MASTER EN SCIENCES INFORMATIQUES

Conception et Implémentation Ergonomiques d'une Application Interactive pour la Construction de Bâtiments Professionnels

Saadaoui, Sanae

Award date:
1991

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique

Rue Grandgagnage, 21, B-5000 NAMUR (Belgium)

CONCEPTION ET IMPLEMENTATION ERGONOMIQUES D'UNE APPLICATION INTERACTIVE POUR LA CONSTRUCTION DE BATIMENTS PROFESSIONNELS

Sanae SAADAOUI

Promoteur: Professeur Jean Ramaekers
Co-promoteur: M.Jean Vanderdonckt
Mémoire présenté en vue de l'obtention
du titre de Licenciée et Maître en Informatique

Année académique 1990-1991

RESUME

Concevoir une interface homme/ordinateur relève de deux domaines: les sciences cognitives et l'informatique.

Ce travail fournit une synthèse des modèles cognitifs et des techniques informatiques qui contribuent à l'interaction entre l'homme et l'ordinateur.

Une deuxième partie de ce travail est consacrée à l'évaluation ergonomique d'une interface qui intègre ces deux disciplines. L'intérêt de cette évaluation est d'une part, de montrer l'importance des connaissances ergonomiques, et d'autre part, leurs manques.

ABSTRACT

Designing a human/computer interface involves two disciplines: cognitive psychology and computer science.

This work provides a synthesis of cognitive theories and computer techniques that contribute in the human/computer interaction.

A second part of this work is dedicated to an ergonomic evaluation of an interface wich integrates the two disciplines. This evaluation is interesting because it shows on one side, the importance of ergonomical rules and on the other side, their drawbacks.

REMERCIEMENTS

Mes premiers remerciements s'adressent à M. Jean Ramaekers et à M. Jean Vanderdonckt qui, par leurs encouragements et leurs conseils, ont contribué à l'aboutissement de ce mémoire.

Je tiens aussi à remercier M. Bernard Peytoureau et M. Bernard Thomas qui, tout au long du stage que j'ai eu le plaisir de passer chez eux, n'ont jamais cessé de me fournir leur soutien et n'ont jamais hésité à m'accorder de leur temps afin de m'aider à résoudre les problèmes que j'ai pu rencontrer.

Enfin, je voudrais exprimer ma gratitude à ma famille et à mes amis pour le soutien et la compréhension qu'ils m'ont témoigné tout au long de mes études.

Table des matières

INTRODUCTION

L'objectif principal d'un concepteur est de fournir un système puissant qui offre toutes les fonctionnalités désirées par l'utilisateur. Le progrès technique et technologique qu'offre aujourd'hui l'informatique interactive permet de faciliter l'utilisation de ce système.

L'utilisateur tout en étant étranger au monde de l'informatique, a pour objectif de réaliser une tâche bien précise. Souvent, il est incapable de comprendre un tel système mais il essaye de s'y adapter. Cet effort intellectuel peut induire un sentiment de frustration, de mécontentement et surtout d'incapacité de la part de l'utilisateur. Il se sent perdu et désorienté et sans aucun contrôle sur le système.

Le problème principal réside dans la communication entre l'utilisateur et l'ordinateur. C'est au niveau de l'interface homme/ordinateur que se passe cette communication, ou plus précisément, interaction. La pauvreté de cette interface est souvent la source d'une mauvaise interaction, qui mène à une sous-utilisation du système et à une non satisfaction de l'utilisateur.

Le propos de ce travail est de montrer tous les enjeux de cette interaction. L'interface doit être conçue de telle manière qu'elle satisfait d'abord aux exigences de l'utilisateur. C'est l'interface qui doit s'adapter aux utilisateurs et non le contraire. L'ordinateur n'est pas seulement un outil intermédiaire. C'est surtout un collaborateur qui exécute les commandes de l'utilisateur mais aussi le guide et l'oriente dans son travail.

Des interfaces *évoluées* ou *conviviales* sont des interfaces concues en tenant compte des espérances de l'utilisateur. Elles sont non seulement plaisantes à utiliser, mais requièrent aussi un effort cognitif minimum.

La conception d'interface évoluées doit dès lors tenir compte des capacités cognitives de l'être humain.

Cette interaction entre l'homme et l'ordinateur sera illustrée par l'étude d'une application interactive pour la construction de bâtiments professionnels cette étude a fait l'objet d'un stage en milieu professionnel.

La première partie de ce travail décrit l'environnement dans lequel s'est effectué le stage et introduit aux différents éléments et objectifs du projet du stage.

La deuxième partie synthétise des modèles et des théories qui constituent un apport des sciences cognitives aux problèmes des interface homme-ordineteur ou en abrégé IHO. On verra:

- le modèle de la tâche
- le modèle des deux golfes
- le modèle du processeur humain

Cette partie sera complétée par quelques règles et recommandations ergonomiques dont le but est d'aider le concepteur dans sa tâche de réalisation d'interfaces ergonomiques et évoluées.

La troisième partie traite des différents éléments et concept qui interviennent dans l'interaction entre l'homme et l'ordinateur le objets interactifs, les moyens d'interaction mais surtout, les styles d'interaction dont on présentera:

- les langages de commandes
- la sélection de menus
- le remplissage de formes
- la manipulation directe

La quatrième partie évalue ergonomiquement l'interface réalisée dans le stage. Cette évaluation sera faite par rapport aux modèles cognitifs et critères ergonomiques qui seront énoncés dans les premières parties.

PARTIE 1: DEFINITION DU PROJET
chap I: Description du projet

Introduction

Cette partie introductive décrit les différents composants du stage: son environnement de travail, son outil, à savoir le logiciel Cyprion, et ses objectifs.

I.1 Contexte organisationnel

La Commercial Intertech est une société américaine de dimension européenne implantée à Diekirch au Luxembourg. Plusieurs centres sont répartis dans différents pays d'Europe: France, Espagne, Allemagne, Norvège, Grande Bretagne (voir fig 1.1). Diekirch est le siège européen ou s'effectuent tous les traitements concernant les Finances, l'Engineering, le marché, le développement et la production.

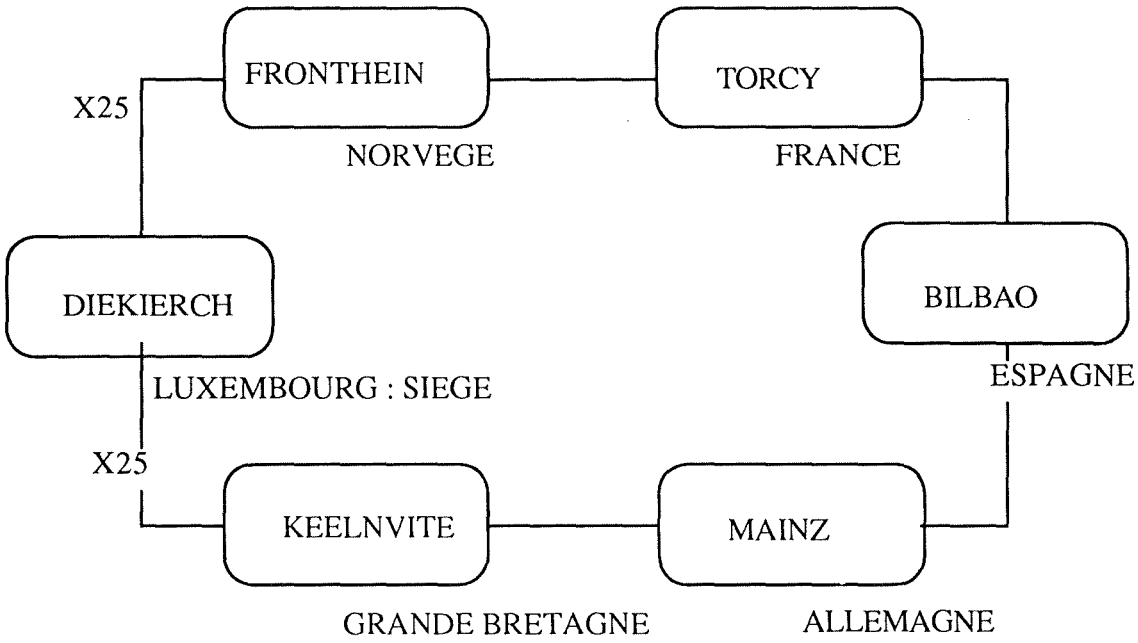


fig 1.1 Le siège de la société et ses centres

La Commercial Intertech fabrique des *bâtiments industriels métalliques* appelés aussi *constructions métalliques* ("Building systems" en anglais). Les bâtiments Astron sont destinés à des secteurs variés: Hall omnisports, hangars d'avions, bureaux, entreprises... Ils différent par le secteur auquel ils sont destinés, par leur forme, leur dimension, leur finition, les charges qu'ils

supportent... Dans les limites des standards définis par la société, chaque construction est adaptée aux exigences et souhaits du client.

Ces bâtiments sont distribués dans toute l'Europe par l'intermédiaire de concessionnaires franchisés et qui sont reliés à l'un des centres.

Les concessionnaires sont en rapport direct avec les clients finals.
Un client qui désire acheter une construction métallique, contacte un des concessionnaires avec qui il définit son *offre* Le concessionnaire calcule le prix du bâtiment et fait une proposition au client.

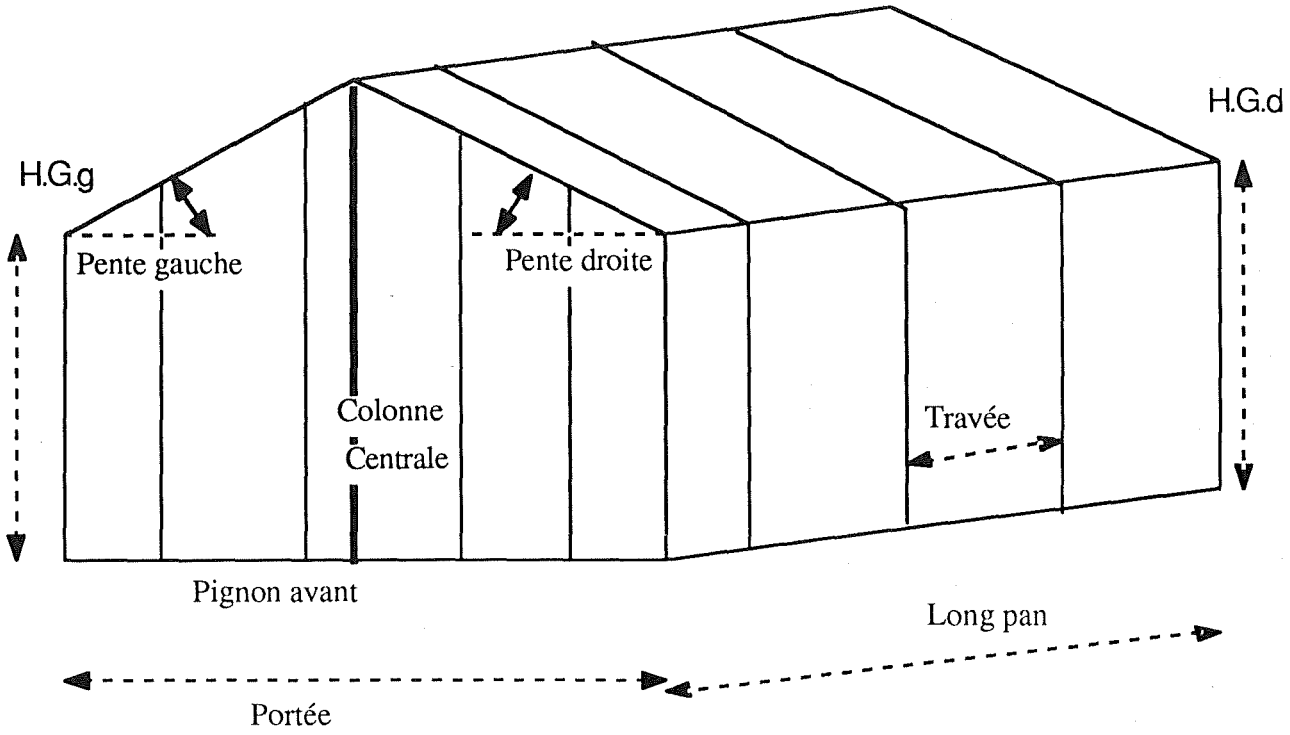
I.2 Infrastructure informatique

Chaque centre dispose d'un ordinateur central (PRIME). Ces ordinateurs sont reliés entre eux par un réseau X.25. Les concessionnaires disposent de micro-ordinateurs de type PC connectés aux ordinateurs centraux via les réseaux de transmission par paquets (Transpac, DATex, PSS, LuxPAC...) appartenant aux pays.

I.3 Description des bâtiments

Une construction métallique ou bâtiment industriel est composée d'un bâtiment principal et d'architecturaux. Les architecturaux sont des *appentis,* des *auvents*, des *acrotères* ou des *saillies de toitures.*

Un bâtiment principal ou un appentis est formé d'une *toiture*, de *murs de pignons* et de *long-pans*. La toiture possède une *pente* gauche et une pente droite (qui peuvent être différents). Les murs de pignons se caractérisent par une *distribution de colonnes* de pignon. Les pignons avant et arrière peuvent avoir des distributions différentes. Sur chaque pignon peut être placée une *colonne centrale*. La largeur des murs de pignons constitue la *portée* du bâtiment. Les long-pans se caractérisent par une *distribution de travées* (voir fig 1.2).



H.G.g : hauteur à la gouttière gauche
H.G.d : hauteur à la gouttière droite

fig 1.2 Un bâtiment

Un bâtiment se caractérise aussi par ses *hauteurs aux gouttières* gauche et droite (qui peuvent être différentes) et une *surface par descente d'eau.*

Un bâtiment est conçu pour supporter une *charge de neige* et une *charge de vent* qui dépendent de sa situation géographique.

Les colonnes peuvent être *autostables* (*autoportantes*).

Les murs et les toitures possèdent une certaine *finition* et un type de *bardage extérieur.* A l'intérieur du bâtiment principal peuvent être placés des *mezzanines*, des *ponts roulants* ou des *monorails* qui constituent une *charge additionnelle* dont il faut tenir compte dans les calculs.

D'autres *accessoires* comme les portes, les fenêtres, peuvent être placées sur les différents types de bâtiments.

Il est aussi possible *d'isoler* les toitures et/ou les murs ou d'y placer du *bardage intérieur.* Des *sous-basements* peuvent être placés sur les murs.

Dans ce qui suit, on appelera *offre* l'ensemble des caractéristiques d'une construction : le bâtiment principal, l'isolation, le bardage intérieur, et tous les paramètres qui s'y attachent.

I.4 Le système actuel:

I.4.1 Description:

Pour faciliter le travail des concessionnaires, un système informatique de chiffrage, appelé ***Cyprion***, a été développé. Ce système produit les résultats suivants:

- le coût du bâtiment
- le coût de transport
- les heures de montage
- la descente des charges pour le calcul des fondations.

Le système a été écrit à l'aide d'un langage de quatrième génération: Pick = Prime information. Il est composé de deux parties:

1. un programme installé sur les PC des concessionnaires et qui gère les communications entre le PC et l'ordinateur serveur central.
2. un logiciel installé sur l'ordinateur central ou tous les traitements sont effectués.

La partie qui nous intéresse ici est celle de l'introduction de données relatives à une offre pour le chiffrage (calcul du prix de l'offre). Le processus d'introduction de données et de chiffrage se déroule en plusieurs étapes (voir diagramme de flux de la figure 1.3) :

- exécuter le programme de communication entre le PC et le serveur
- établir la connexion avec l'ordinateur central via la régie des téléphones
- identifier l'utilisateur
- exécuter le programme d'introduction des données
- effectuer le chiffrage
- afficher les résultats
- terminer la communication et la connexion.

I.4.2 Critique de l'existant

Deux remarques sont à faire sur le processus d'introduction des données et de chiffrage décrit plus haut :

1. le temps de communication entre le PC du concessionnaire et l'ordinateur serveur est le temps écoulé entre l'établissement et la fin de la connexion. Donc un temps assez long.
2. toutes les données introduites, les écrans, les textes des menus et les résultats du chiffrage transitent par le réseau. Donc, une grande charge transférée.

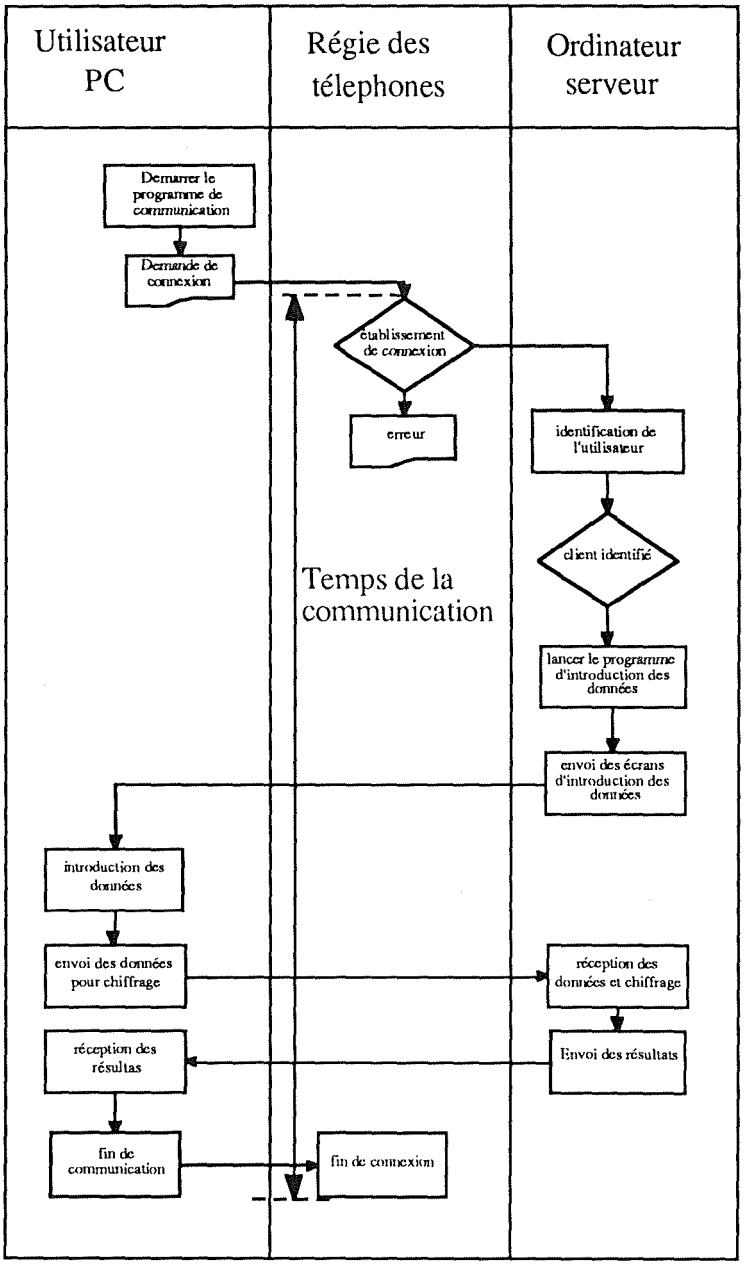


fig 1.3 Diagramme de flux du processus d'introduction des données

Le réseau utilisé est un réseau de transmission par paquets. Le coût de transfert dépend donc de la durée de la communication et de la charge transférée. d'après les deux remarques, le coût de transmission s'avère assez élevé. Il augmentera en fonction du taux d'erreurs. celles-ci impliqueront l'envoi de messages d'erreur, l'introduction de nouvelles valeurs, de nouvelles validations, ce qui allongera la communication et accroîtra la charge du réseau.

D'un autre côté, l'interface d'introduction des données est une succession simple de menus sans aucune considération ergonomique (voir exemple fig 1.4). Il en découle un manque de souplesse au niveau de l'exécution de la tâche ainsi qu'un accroissement de la complexité du travail.

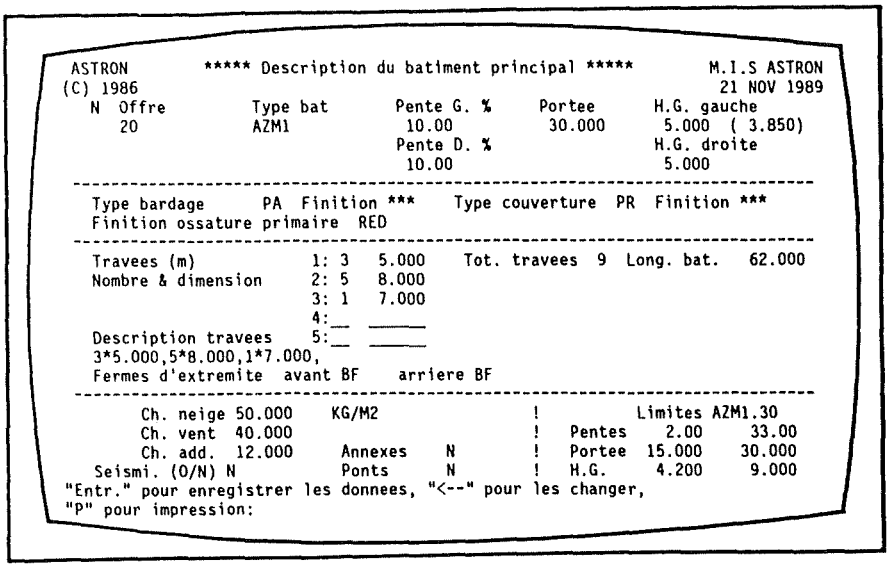


fig 1.4.Exemple d'un écran

En plus, des limitations contraignent fortement l'introduction de certaines données telles que l'isolation et le bardage intérieur. "il n'est pas possible d'isoler une toiture partiellement ou d'introduire différents types d'isolation". De plus, "les murs du bâtiment principal ou des appentis ne peuvent pas être isolés partiellement ou individuellement".(ref: manuel d'utilisation).

La partie concernant le bardage intérieur n'a pas été développée, ce qui empêche toute prise en compte des données de celui-ci.

I.5. Objectifs du projet

Les deux principaux objectifs du projet sont d'une part la réduction des coûts de transmission des données sur le réseau à commutation par paquets, et d'autre part la refonte en un système plus convivial, plus ergonomique et plus facile à utiliser.

Un objectif annexe consiste à lever certaines contraintes:

- permettre l'isolation partielle et individuelle des murs et des toitures
- permettre l'introduction du bardage intérieur et, comme pour l'isolation, permettre un bardage partiel.

I.6. Solutions proposées

Pour réduire les coûts de transmission, toute la partie d'introduction des données sera écrite sous forme de programme résident sur les PC des concessionnaires. Autrement dit, un concessionnaire introduit ou modifie les données relatives à une offre localement sur son propre PC. La connexion du PC avec l'ordinateur central n'est établie que pour l'envoi de données à chiffrer et la réception des résultats du chiffrage. Le nouveau processus est décrit par la figure 1.5

Pour réduire la charge des données transmises, seules les valeurs des données strictement nécessaires au chiffrage seront envoyées sous une forme compactée au serveur. Un programme de conversion les remettra sous leur forme exploitable. Le même processus est utilisé pour l'envoi des résultats au PC. Ceci est illustré par la figure 1.6

Pour rendre le système plus convivial, une interface homme/ordinateur évoluée sera étudiée et mise au point . Sa conception reposera sur des considérations ergonomiques reconnues et utilisera des techniques de manipulation nouvelles comme la souris.

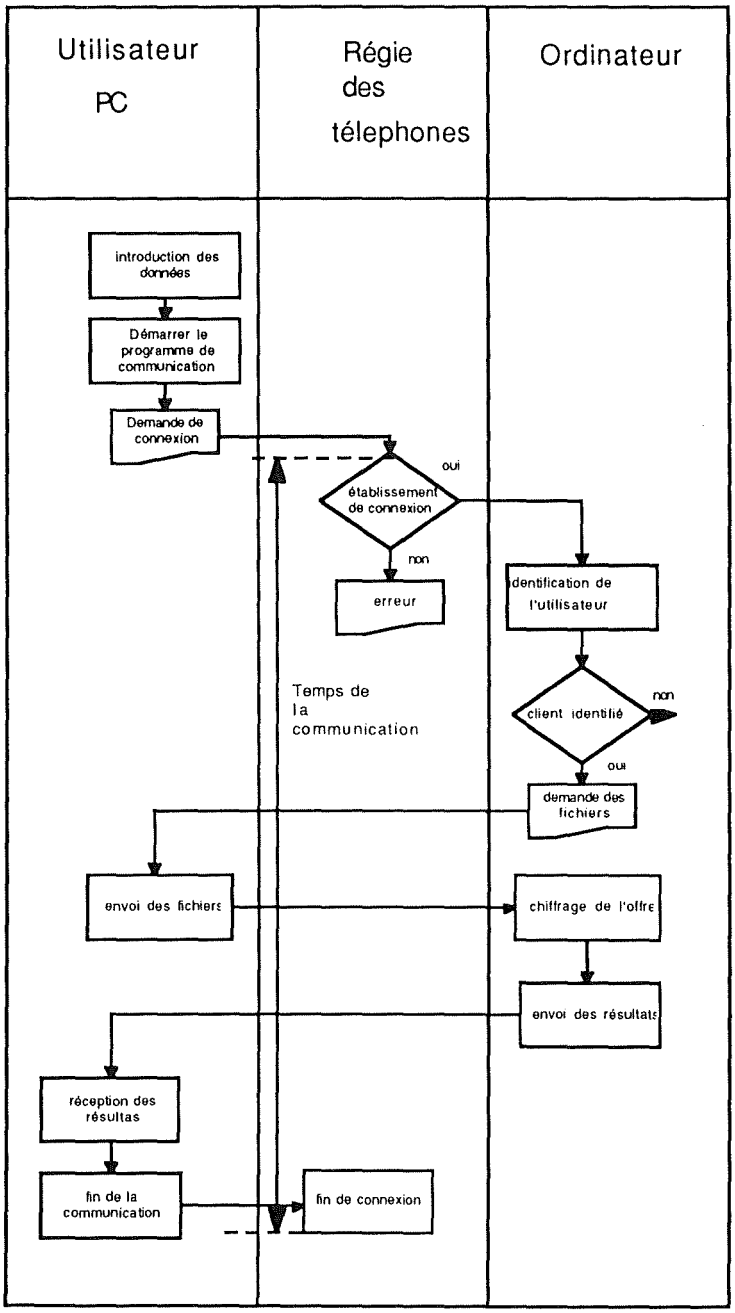


fig 1.6 Diagramme de flux du nouveau système

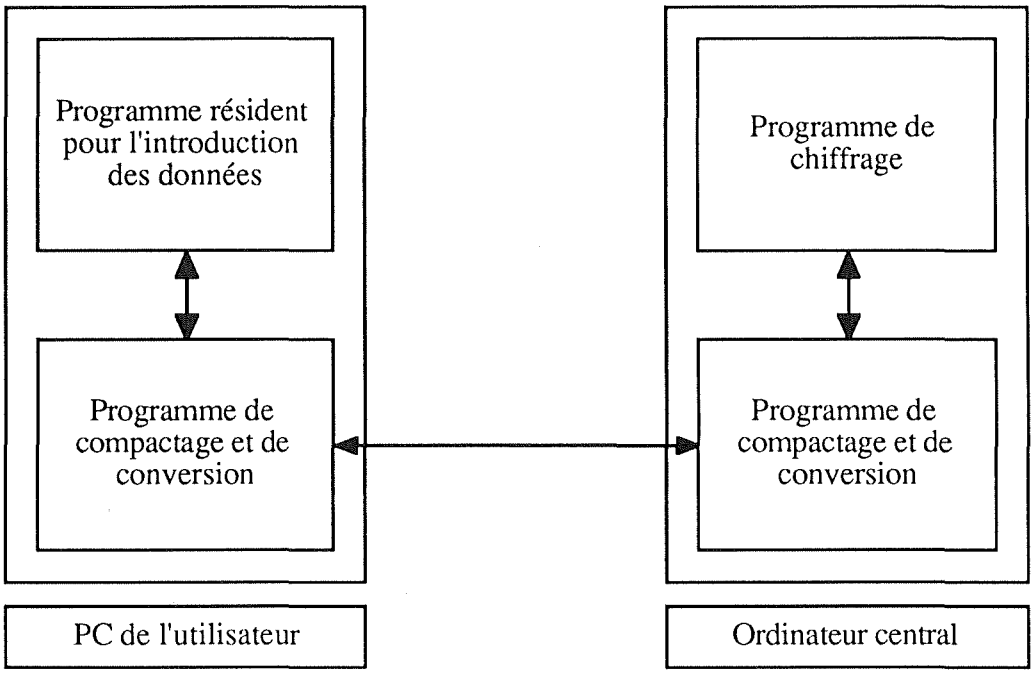


fig 1.5 Echange des données sous forme compacte

Le langage utilisé pour réécrire la partie d'introduction des données est le TURBO C (voir Borland) avec utilisation du mode graphique VGA pour visualiser le projet.

I.7. Evaluation des résultats

L'interface d'introduction des données est résidente sur les PC des concessionnaires. Le programme de chiffrage se fait toujours sur l'ordinateur serveur. La comparaison des deux diagrammes de flux respectivement de l'ancien et du nouveau système, permet de constater la diminution considérable du temps de communication entre le PC du concessionnaire et l'ordinateur serveur.
D'un autre côté, l'utilisation de fichier compactés pour le transfert de données strictement nécessaires aux calculs, réduit énormément la charge d'informations transmises sur le réseau.
Ceci permet d'éviter aux concessionnaires les problèmes liés à l'établissement de la connexion.
Du point de vue économique, les coûts relatifs à la communication sont réduits.

Dans ce qui suit, une critique et une évaluation de l'interface réalisée seront exposées. Elles seront faites par rapport aux critères et régles

ergonomiques cités par différents auteurs et chercheurs dans le domaine de conception d'interfaces homme/ordinateur évoluées.

PARTIE 2: SCIENCES COGNITIVES

La prise en compte des facteurs cognitifs de l'être humain dans la réalisation et la conception des systèmes interactifs est devenue un fait fondamental.
Les connaissances intuitives ne sont plus suffisantes. La difficulté des systèmes actuels croit de manière continue et rapide. Il est devenu impératif d'essayer de comprendre les principes et mécanismes qui régissent les actions et comportements de l'être humain en tant qu'un utilisateur de tels systèmes. Cette connaissance utilisée judicieusement contribue à l'amélioration de l'interaction homme/machine.
Les deux chapitres suivants décrivent ces connaissances cognitives selon deux approches: une **approche théorique** et une **approche expérimentale.** La première aide le concepteur à mieux comprendre et à déterminer les besoins de l'utilisateur, alors que la deuxième le guide dans la conception de l'interaction.

Chapitre II : Quelques modèles

Introduction :

Dans ce chapitre seront décrits trois modèles théoriques qui constituent un apport des sciences cognitives à la conception des systèmes interactifs:

1. le modèle de la tâche
2. le modèle des deux golfes
3. le modèle du processeur humain

Les deux premiers modèles analysent les processus psychologiques qui mènent à un comportement de l'individu. S'inscrivant dans la théorie de l'action de D.Norman [Norman 86] ils décrivent de manière informelle les paramètres qui interviennent dans la réalisation d'une tâche.
Le modèle du processeur humain de S.Card, T. Moran et A.Newell [Card,83], décrit par contre, le comportement humain de manière formelle. Il utilise une terminologie similaire à celle qu'utilise l'informaticien.

II.1 Le modèle de la tâche

II.1.1 Variables physiques et variables psychologiques

Une personne veut réaliser une *tâche*. Elle a des objectifs et des intentions qu'elle exprime en termes psychologiques. Ce sont des *variables psychologiques* définies dans un univers psychologique. Pourtant, la tâche sera réalisée sur un système physique, défini par des variables d'état. Ce sont les *variables physiques*. Le changement de l'état du système, donc de ses variables d'état, se fait par la manipulation de certains mécanismes physiques.

Exemple:

L'utilisateur a pour objectif de déplacer le curseur vers une position déterminée. C'est une intention, une pensée qu'il a dans sa tête: c'est une variable psychologique. Pour réaliser son objectif, il doit traduire cette intention en action. Ce qui se traduit par la manipulation d'une souris. C'est le système physique. La souris est caractérisée par une vitesse de déplacement et une direction, ce sont les variables physiques.

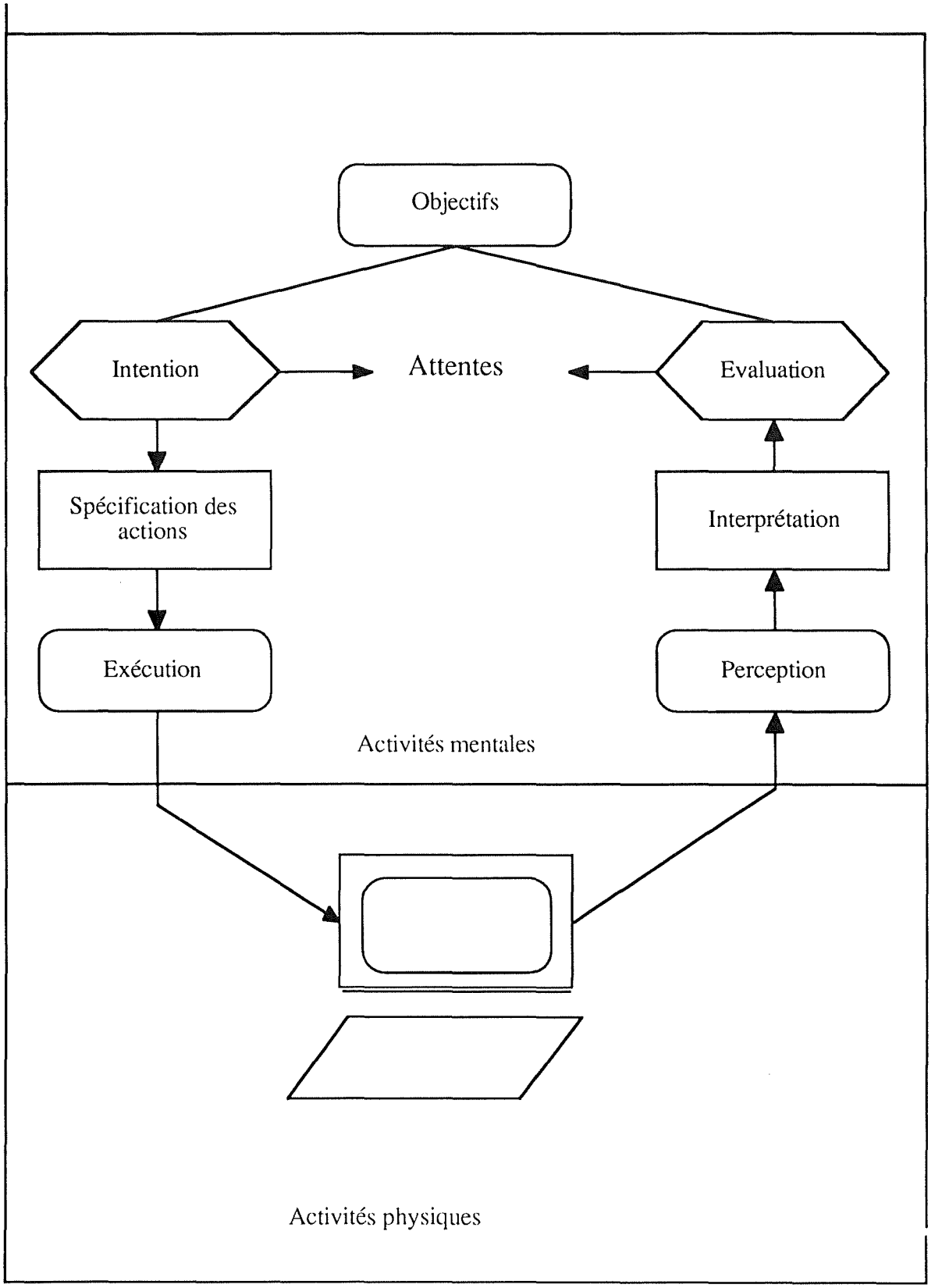
II.1.2 Complexité d'une tâche

Les tâches varient par leurs natures et leurs complexités. Cette complexité est fonction du nombre de paramètres et de variables à contrôler. Mais , une tâche avec une seule variable n'est pas nécessairement simple. Cette difficulté peut provenir d'une mauvaise formulation ou d'une mauvaise modélisation de la tâche.
L'utilisation d'un modèle adéquat et un choix judicieux des mécanismes de contrôle contribuent à diminuer la complexité d'une tâche.

II.1.3 Les aspects d'une tâche

Les notions de contrôle, de variables et d'évaluation font partie de l'ensemble des aspects d'une tâche. Une tâche met en jeu sept activités (voir fig 2.1) :

1. établir un but : c'est à dire définir l'état du système qu'on souhaite atteindre.
2. spécifier une intention : l'intention est la décision d'agir pour réaliser un but
3. spécifier la séquence d'actions
4. exécuter la séquence d'actions
5. percevoir l'état du système
6. interpréter l'état du système
7. évaluer l'état du système par rapport aux buts et intentions.



voir fig 2.1 activités d'une tâche
(tiré de l'article de D.Norman)

Ces activités peuvent ne pas apparaître dans l'ordre. Elles peuvent être répétitives. Quelques-une peuvent être évitées. Tout dépend de la tâche et de sa complexité.
Dans certains cas la personne est réactive, elle répond aux événements. Cette réaction peut contredire les buts et les intentions fixés au début.

Exemple:

Une personne définit un appentis. Elle analyse le graphique affiché et décide qu'il y a des modifications à faire car il ne correspond pas à ses attentes.
La personne établie alors une première intention I1, qui est "changer la position de l'appentis". I1 ne dit pas ce qu'il faut faire au juste. La personne émet alors une deuxième intention plus concerte, I2 "déplacer l'appentis sur la première travée". Pour réaliser I2 une séquence d'actions est générée puis exécutée. (voir fig 2.2)

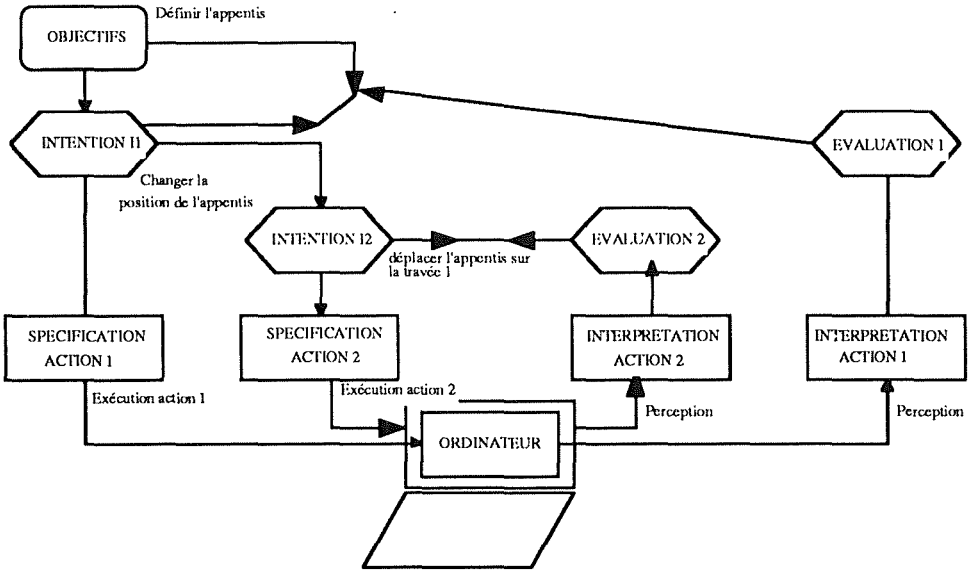


fig2.2 Exemple de la décomposition d'une tâche

L'intérêt de cette classification est de décomposer successivement une tâche en sous-tâches plus simples, jusqu'à obtenir des tâches élémentaires facilement réalisables. Ce qui revient à dire, diminuer les divergences entre les variables psychologiques et les variables physiques.

II.2.Modèle des deux golfes

Le problème concerne la traduction des variables psychologiques en variables physiques et réciproquement. Ce problème sera d'autant plus difficile à résoudre que les écarts entre ces variables sont grands.
Le modèle des deux golfes représente ces divergences sous formes de deux golfes (deux fossés) entre lesquels deux ponts sont jetés.

- le pont pour le *golfe d'exécution*
- le pont pour le *golfe d'évaluation*

II.2.1 Golfe d'exécution et golfe d'évaluation :

Le système physique et les objectifs forment deux univers différents, l'un physique l'autre psychologique. Ils ont des formes différentes et des contenus différents. Ceci crée deux golfes (voir fig 2.3). Un premier golfe d'exécution se situe entre les objectifs et les variables d'actions du système physique. Il exprime la difficulté à traduire les objectifs en termes physiques. C'est à dire en actions exécutables pour atteindre le but recherché. Un deuxième golfe, d'évaluation est situé entre les variables d'état et l'évaluation du système. Ce qui signifie, qu'ayant un certain état du système, résultat de certaines manipulations comment peut-on évaluer et traduire cet état pour décider si on a atteint oui ou non les objectifs fixés.

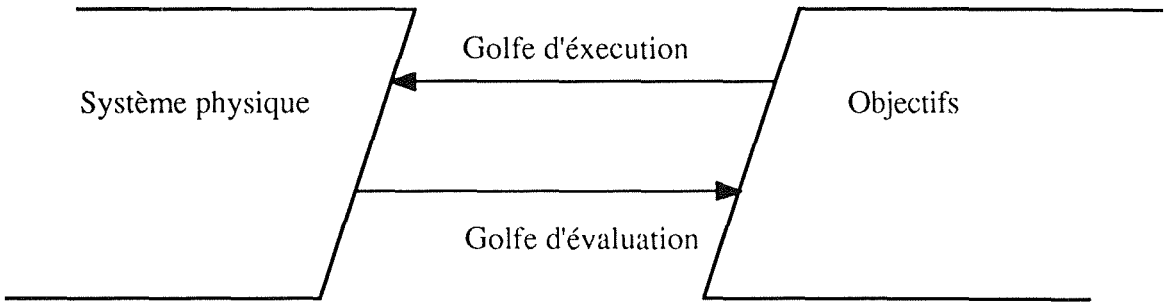


fig 2.3 Les golfes d'exécution et d'évaluation

Des ponts sont jetés entre ces deux golfes. Ils peuvent être établis de deux façons différentes. Soit partir du système physique et approcher les objectifs de la personne. Soit créer des plans, des séquences d'actions et des interprétations qui approchent la description des objectifs de la description requise par le système.

II.2.1.1 Construction du pont d'exécution

Le pont d'exécution est établit de telle sorte que les commandes et mécanismes du système correspondent le plus possible aux pensés et objectifs de l'utilisateur (voir fig 2.4). Ceci en quatre étapes:

1. formation de l'intention à partir des objectifs.
2. spécification de la séquence d'actions à exécuter: ce n'est pas chose évidente. L'utilisateur doit traduire ses objectifs et intentions en un état souhaité du système. Il doit déterminer les changements à effectuer aux variables d'état du système pour le mener à l'état souhaité. Il doit déterminer les mécanismes de contrôle nécessaire et les manipulations à appliquer à ces mécanismes. Le résultat est une spécification mentale des actions à exécuter.
3. exécution de la séquence d'actions spécifiée: c'est la première étape physique non mentale. Des actions peuvent être plus difficiles à exécuter que d'autres. Le choix des dispositifs et des moyens de saisie peut influencer la sélection d'actions et par suite la différence entre le système et les intentions.
4. établir le contact avec les mécanismes d'input de l'interface.

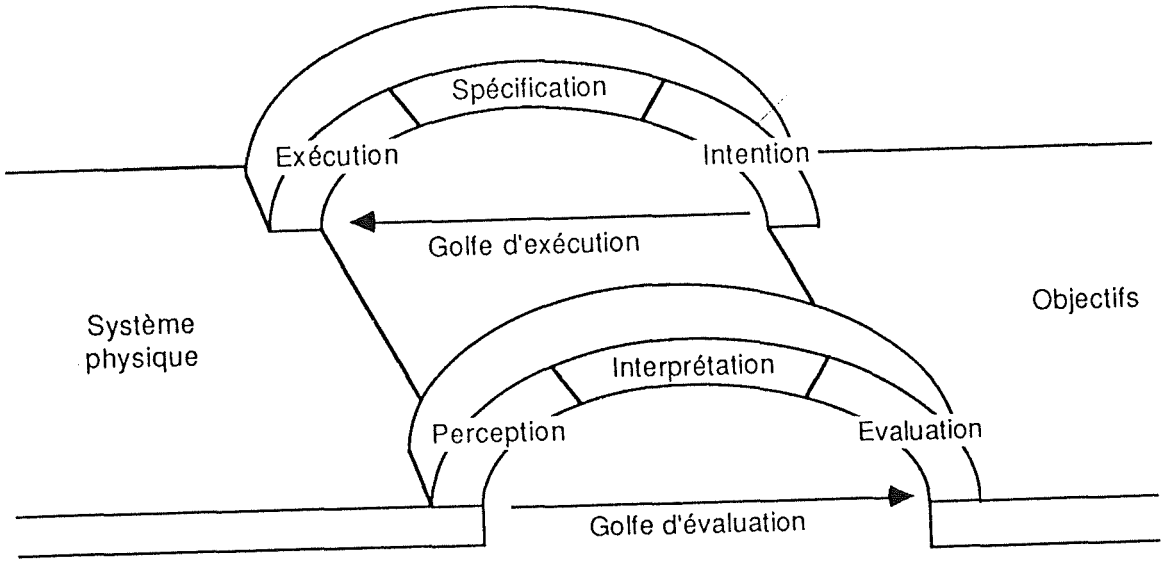


fig 2.4 Ponts pour le golfe d'exécution et d'évaluation

Exemple

Soit un utilisateur dont l'objectif est de modifier la distribution des colonnes du pignon avant d'un appentis. Le processus d'établissement du pont d'exécution se passe de la manière suivante:

1. formation de l'intention: l'utilisateur décide de modifier la portée pour modifier le nombre des colonnes du pignon avant de l'appentis.
2. spécification des actions:
   - sélectionner le champ correspondant à la portée.
   - modifier la valeur de la portée
3. exécution des actions avec l'aide de la souris et du clavier.
4. affichage des résultats sous forme d'un graphique.

II.2.1.2 Construction du pont d'évaluation

Les résultats du système sont affichés par le système lui-même. Ces affichages sont exprimés par des variables physiques liées aux variables psychologiques. Les affichages déterminent l'état du système. L'évaluation consiste à comparer l'interprétation de l'état du système aux objectifs fixés au début par l'utilisateur. C'est-à-dire voir si l'état du système correspond à l'état désiré.

Le pont est établi en quatre étapes (voir fig 2.4):
1. affichage des résultats de l'interface
2. perception des affichages
3. interprétation des résultats
4. évaluation des résultats.

Exemple: établir le pont pour le golfe d'évaluation de l'exemple précédent:

1. affichage du graphique output de l'interface
2. perception des résultats: percevoir la distribution des colonnes du pignon avant
3. interpréter les résultats: voir si la distribution des colonnes correspond à celle qui était espérée.
4. évaluer: but atteint ou non.

Il peut y avoir plusieurs niveaux de résultats qui correspondent à différents niveaux d'intentions. Ceci pose un problème. La réponse du système doit être immédiate après l'exécution des actions sinon, l'utilisateur risque d'oublier à quelle intention correspond la séquence d'actions exécutée et ainsi rendre impossible l'évaluation.

Le but recherché par la construction des ponts est la réduction de *l'effort cognitif* requis pour la manipulation et l'évaluation du système. C'est à dire la réduction des distances entre l'univers psychologique dans lequel l'utilisateur exprime ses objectifs et l'univers physique dans lequel sera exécuter la tâche.

Plus les variables physiques (i.e. du système) sont proches des variables psychologiques (i.e. celles de l'utilisateur), plus les ponts sont courts et plus l'effort cognitif de transposition sera minimisé.

La qualité des ponts et la réduction des distances va de ce fait déterminer la qualité de l'interface entre l'utilisateur et le système.

II.3. Modèle du Processeur Humain

Ce modèle voit l'être humain comme un système de traitement d'information.
Comme pour un système d'information, le modèle du processeur humain peut être décrit par un ensemble de mémoires et de processeurs.

Le modèle du processeur humain est composé de trois sous-systèmes: le système sensoriel, le système moteur et le système cognitif. Ces derniers peuvent être interdépendants ou isolés. Chacun de ces sous-systèmes possède ses propres mémoires et processeurs.

Les mémoires peuvent être décrites par les paramètres suivants:

m, *la capacité de la mémoire*, c'est-à-dire le nombre d'items mémorisables par la mémoire

d, *la persistance de l'information* : représente le temps au bout duquel la probabilité de restituer un item est supérieur à 0.5

k, *le type de codage* (physique, symbolique, sémantique, acoustique)

Les processeurs sont caractérisés par :

t, *cycle de base du processeur*. Ce cycle inclut le temps d'accès à sa mémoire locale.

II.3.1 Le système sensoriel

Le système sensoriel détecte des sensations en provenance du monde physique, les stimuli et les transforme en une représentation codée, interne au système.

Le système sensoriel consiste en un ensemble de sous-systèmes dont chacun est responsable d'une classe de stimuli. A chacun de ces sous-systèmes est associée une mémoire sensorielle. Les plus intéressantes de ces mémoires sont les mémoires du *sous-système auditif* et *visuel*, *Visuel Image Store* (VIS) et *Auditif Image Store* (AIS).

Les stimuli sont codés de manière non symbolique. Seules leurs propriétés physiques sont prises en compte. Leur signification ne relève pas du domaine du système sensoriel.

Exemple de l'oeil :

L'oeil est sensible à la lumière. Si un rayon de lumière frappe la rétine de l'oeil, celui-ci va enregistrer l'intensité et la longueur d'onde de la lumière, ce qui est représenté par le type de codage k_S.

k_{ais} = physique

k_{vis} = physique.

Si l'oeil voit la lettre A par exemple, il va enregistrer les courbatures et la forme de la lettre.

La reconnaissance de la représentation symbolique d'un stimulus s'effectue dans la mémoire du système cognitif (voir fig 2.5). En effet, il existe un lien étroit entre les mémoires sensorielles et le mémoires cognitives.

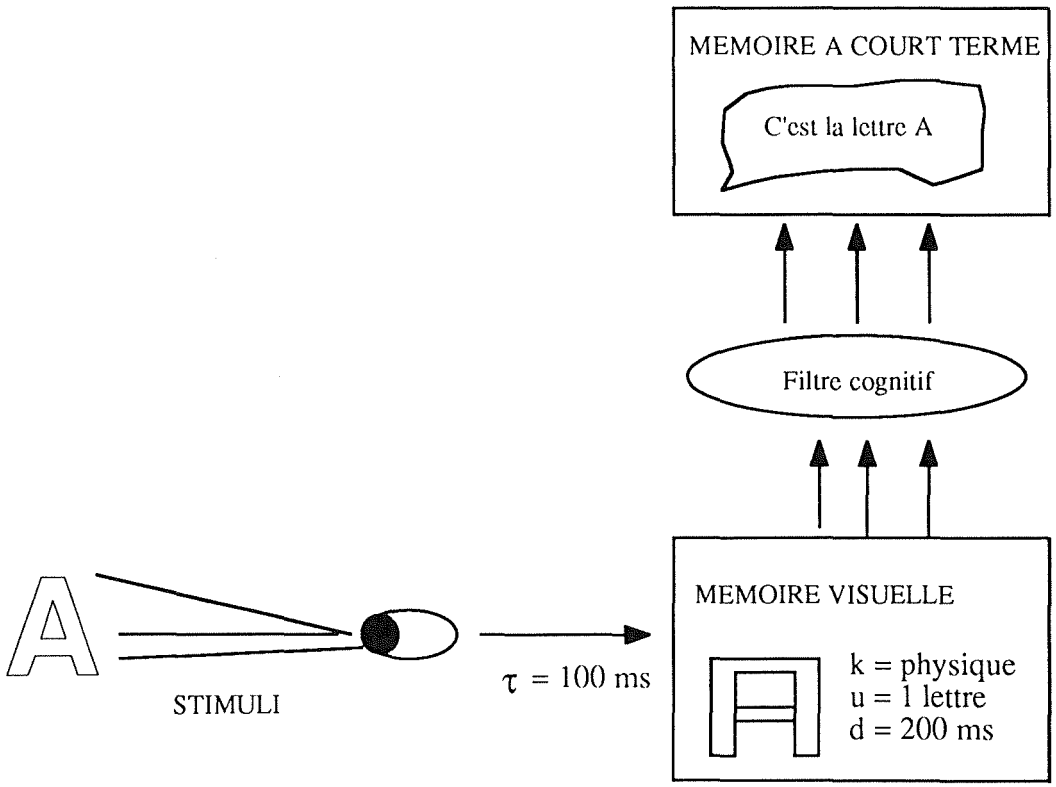


fig 2.5 Le système sensoriel visuel et sa relation avec le système cognitif.

Le système cognitif peut spécifier quelles parties des mémoires sensorielles seront codées. Cette spécification ne peut être que physique car la seule disponible.

Exemple

Si une liste colorée de lettres est présentée à un utilisateur, il peut ne sélectionner qu'une partie, par exemple celle dont les lettres ont une couleur déterminée et qui se trouvent dans la partie supérieur du VIS. Le système cognitif dispose pour cela d'un filtre cognitif.

La persistance des mémoires sensorielles est de l'ordre de 200 msec pour la mémoire visuelle et de 1500 msec pour la mémoire auditive.

Le cycle de base t_S du système sensoriel est de l'ordre de 100 msec.

C'est le temps écoulé entre la réaction du système sensoriel à un stimulus et le moment ou celui-ci est représenté dans une mémoire sensorielle.

Deux événements sensoriels qui surviennent en même temps durant un même cycle sont combinés en un seul. Si l'oeil voit deux fois la lettre A pendant le même cycle, les deux stimuli sont combinés en un seul plus condensé.

Le cycle de base du processeur n'est pas constant, il varie suivant les conditions. Il est plus court quand le stimulus est plus intense. Le cycle de base Ts du processeur varie inversement avec l'intensité du stimulus.

II.3.2 Le système moteur

Les pensés son traduites en mouvements par l'activation de certains muscles. Les mouvements les plus intéressants pour l'interaction homme/ordinateur sont les mouvements de manipulation des unités physiques de commandes, telles que le clavier, la souris.

Ces mouvements ne s'effectuent jamais continûment, mains sous la base de suites de micromouvements discrets.

Le cycle de base du processeur moteur est de 70 mse

C'est le temps que met un micromouvement pour s'accomplir.

Le temps théorique pour placer la main sur une cible donnée est :

$T_m = I_m * \log2 * D/L$, loi de Fitts

où (voir fig 2.6),

D, distance à parcourir par la main

L, longueur de la cible

I_m, constante.

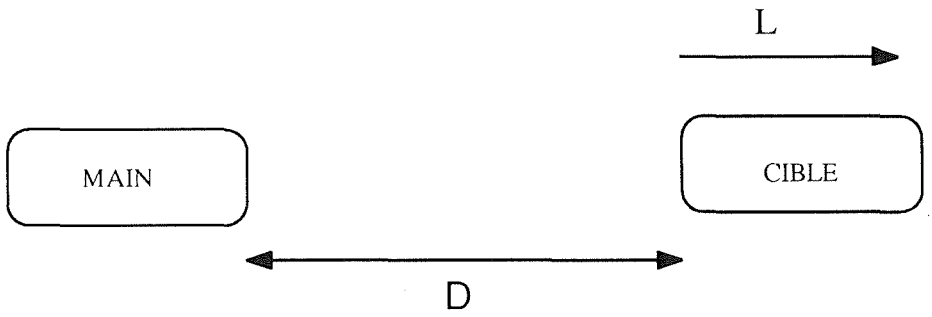


fig 2.6 Micromouvements effectués par la main dans son déplacement vers la cible

Pour atteindre la cible, la main effectue une succession de micromouvements qui sont des petits déplacements.

II.3.3 Le système cognitif

Pour des tâches simples, le système cognitif fonctionne comme un intermédiaire entre le système moteur et le système sensoriel. Il connecte les entrées aux bons résultats. Mais en général, les tâches sont complexes et nécessitent de la connaissance et la résolution de problèmes.

Le système cognitif comprend deux mémoires. La **mémoire à court terme**, ou mémoire de travail, elle contient les informations en cours de manipulation. La **mémoire à long terme** elle, contient les connaissances permanentes. Celles-ci pourront être utilisées par la suite dans la mesure de l'aptitude de l'être humain à restituer ces informations.

Fonctionnellement, la mémoire à court terme est l'endroit où les opérations mentales obtiennent leurs opérandes et laissent leurs résultats. Les opérandes proviennent des mémoires sensorielles ou de la mémoire à long terme. Elle constitue le registre du processeur cognitif et se comporte comme les registres d'un calculateur.

Structurellement, c'est un sous-ensemble des éléments de la mémoire à long terme qui ont été activés.

La mémoire à court terme reçoit du système sensoriel les informations codées symboliquement. Le codage prend en considération la signification réelle du stimulus et non pas ses caractéristiques physiques.

Les éléments activés de la mémoire à long terme forment des unités cognitives symboliques appelées des **mnèmes** (ou "chunks" en anglais). Ces mnémes peuvent s'organiser pour former d'autres mnèmes. La signification d'un mnème dépend davantage du contenu de la mémoire à long terme de l'utilisateur que de la tâche. Par exemple, H.G.g peut signifier Hauteur à la Gouttière gauche pour ceux qui connaissent le sigle. Pour d'autres, qui sont étranger au domaine des bâtiment industriels, ce sera l'association de trois mnèmes car H.G.g ne représente plus une seule unité cognitive.

Un mnème peut être associé à d'autres mnèmes. Dés lors, l'activation d'un mnème entraînera l'activation de tous les mnèmes qui lui sont associés. Un mnème activé devient disponible dans la mémoire à court terme.

La capacité de la mémoire à court terme est de l'ordre de 7+-2 mnèmes

(formule de Georges Miller).

Au delà de ce seuil, on remarque un phénomène d'interférences: la mémoire est saturée. Les nouveaux mnèmes effacent les anciens, ce qui provoque une dégradation au niveau de la rétention et du traitement de l'information.

La mémoire à long terme est constituée d'un réseau de mnèmes associés et accessibles par associativité à partir des contenus de la mémoire à court terme. Elle contient des faits et des procédures, appelés respectivement *connaissances factuelles* ou *connaissances de faits*, et *connaissances procédurales* ou *savoir-faire*.

Les informations du système sensoriel avant d'être enregistrées dans la mémoire à long terme sont codées symboliquement. Le type de codage utilisé est le codage sémantique, k_{mlt} = sémantique. Ceci est important pour représenter les liens qui existent entre les mnèmes. L'opération qu'on vient de décrire est l'opération d'écriture dans la mémoire à long terme.

La manière dont les mnèmes et les contenus actuels de la mémoire à court terme sont codés déterminent le chemin à suivre, en terme d'association, pour retrouver le mnème cible.

Exemple:

Supposons un utilisateur nommant son fichier de travail "brillant" par opposition à "sombre". Si ultérieurement il recherche ce fichier en pensant à "brillant" comme synonyme d'"excellent" (travail brillant), il ne reconnaîtra pas le fichier car il n'utilise pas le même ensemble de mnèmes et donc pas les mêmes associations.

La recherche d'un mnème , qui est l'opération de lecture dans la mémoire à long terme, peut échouer pour deux raisons:

1°) L'association n'est pas trouvée à cause d'interférences de plusieurs mnèmes avec le mnème recherché. Ces interférences peuvent survenir si des mnèmes différents ont été enregistrés avec les mêmes critères d'associations.

2°) A cause des interférences, l'information peut devenir fonctionnellement perdue. C'est à dire, elle sera toujours potentiellement présente mais inaccessible. La persistance de la mémoire à long terme est infinie.

Si le mnème est retrouvé, il est alors activé et transféré dans la mémoire à court terme.

Le processeur cognitif fonctionne selon le cycle "Reconnaissance/Action", analogue au système de production des ordinateurs. Le processeur détermine les actions de la mémoire à long terme associées aux mnèmes de la mémoire à court terme, pendant la phase de la reconnaissance. Les actions vont modifier le contenu de la mémoire à court terme dans la phase d'exécution.

Le cycle de base du processeur cognitif est de l'ordre de 70 msec.

Il est plus petit quand un plus grand effort est induit par un plus grand nombre de demandes ou une grande charge d'information.

Le système cognitif est parallèle dans sa phase de reconnaissance et sériale dans sa phase d'exécution. Il peut être conscient de plusieurs événements en même temps, mais ne peut en exécuter qu'un seul à la fois.

II.4 Conclusion

Le modèle de la tâche et le modèle des deux golfes mettent l'accent sur les distances qui existent entre les variables physiques et les variables psychologiques. Plus ces distances sont grandes plus l'utilisateur doit fournir un effort cognitif considérable pour réaliser sa tâche.

L'objectif de ces deux modèles est de réduire ces distances pour améliorer la qualité de l'interface.

- Le modèle de la tâche décompose la tâche en sous-tâches plus simples.
- Le modèle des deux golfes construit des ponts entre les golfes qui représentent ces distances.
- Le modèle du processeur humain montre les limites des capacités humaines qu'un système d'information ne doit pas dépasser. L'ordinateur est perçu comme une prolongation des capacités humaines: si la mémoire à court terme est limitée, elle sera supplée par celle de l'ordinateur.

Le concepteur doit prendre en considération ces limites lors de la conception de l'interface.

Chap III Considérations ergonomiques

Introduction

Tout d'abord, qu'est ce que l'ergonomie?

"L'ergonomie est une discipline dont l'objet est l'étude du travail de l'homme. Son objectif est l'adaptation du travail à l'homme" (G.Karmas).

Il existe différents types d'ergonomies. Celle qui nous intéresse ici est *l'ergonomie cognitive*. Son objectif est l'amélioration de la qualité des interfaces homme/ordinateur (en abrégé IHO ou IHM pour homme/machine) par la prise en considération des caractéristiques cognitives de l'homme. Comme on a vu dans le modèle du processeur humain, ces caractéristiques concernent sa capacité de perception, de mémorisation, d'apprentissage, sa vitesse de réaction devant une situation particulière.
Comme vu précédemment, la qualité d'une interface est d'autant meilleur que les distances entre les variables physiques et les variables psychologiques sont réduites. C'est à dire que l'interface est d'autant meilleure qu'elle ne requiert pas un grand effort cognitif à l'individu. Reste à savoir comment évaluer la qualité de l'interface pour décider si elle nécessite de la part de l'utilisateur un grand effort cognitif ou non. Cinq critères d'évaluation de la qualité d'une interface seront abordés [Sneiderman,87].

Dans ce chapitre, seront citées d'abord les critères de qualité d'une interface. Puis seront considérées les régles ergonomiques empiriques décrites par [Shneiderman,87] et [Scapin,87]. Ce sont des régles issues de l'expérimentation. Ces régles constituent un point de référence pour la conception d'une interface ergonomique et conviviale. Ensuite seront énoncées quelques recommandations dont le but est de guider le concepteur dans sa tâche de réalisation d'une interface qui satisfait aux régles empiriques. Ces recommandations concernent :

- les saisies
- les affichages
- les écrans
- les dialogues.

Les régles ainsi que les recommandations prennent en compte les limitations des capacités de traitement de l'être humain.

Parce que ces règles et recommandations sont nombreuses Scapin et [Shneiderman,87], nous nous concentrerons uniquement sur celles qui sont pertinentes pour le projet en question.

III.1 Critères de qualités d'une interface

Evaluer la qualité d'une interface, c'est mesurer son degré de convivialité. Cette évaluation analyse l'effort cognitif que fournit l'utilisateur pour contribuer dans la réalisation de la tâche.

Schneiderman propose cinq critères:

1) le temps d'apprentissage: temps nécessaire à l'utilisateur pour apprendre les commandes nécessaires à la réalisation de la tâche.
2) la rapidité de l'exécution: temps nécessaire à la réalisation de la tâche.
3) le taux d'erreurs commises: ce critère concerne,
   - la fréquence des erreurs: les erreurs sont soit des erreurs d'exécution soit des erreurs d'intentions suite à la sélection d'une commande inappropriée.
   - le temps de correction des erreurs.
4) la période de rémanence: période au bout de laquelle l'utilisateur commence à perdre la connaissance qu'il a acquise. Elle est fonction du temps d'apprentissage et de la fréquence d'utilisation.
5) la satisfaction subjective: traduit la satisfaction de l'utilisateur à utiliser le système.

Respecter tous les critères n'est pas toujours possible. Le concepteur se trouve souvent dans une situation de compromis: en fonction de l'application et des utilisateurs, il peut sacrifier un critère au profit d'un autre. Par exemple, négliger la gestion des erreurs au profit d'une exécution rapide si l'interface est destinée à des experts.

III.2 Règles ergonomiques empiriques

Shneiderman a décrit huit règles dont on ne retiendra que celles utiles pour l'interface réalisée. Elles sont: la cohérence, la concision, le retour d'information, la gestion d'erreur et la flexibilité.

R1 la cohérence

Assurer une homogénéité entre les différents composants de l'interface à tout moment de la conception. Elle repose sur le caractère unitaire de ses composants.

R11 Cohérence et spécification du plan des actions

Que ce soit au niveau inter-application ou au niveau intra-applications, la réalisation d'une même sous-tâche doit nécessiter la même suite de commandes ou d'actions.

R12 Cohérence et exécution

Un objet ou un concept doit être désigné par le même nom quelque soit sa position dans les différentes sous-tâches, c'est à dire indépendamment du contexte. De plus, il faut essayer dans la limite du possible, qu'il y ait une cohérence entre les dénominations et le vocabulaire de l'utilisateur.

Un autre point important est la cohérence spatiale: choisir une organisation spatiale adéquate pour les écrans. Par exemple, les messages d'indication seront affichés au même endroit en bas de l'écran. Ceci permet d'automatiser de manière cohérente certaines procédures et aide l'utilisateur à localiser un type d'information.

R2 La concision

Comme vu dans le modèle du processeur humain, la mémoire à court terme de l'opérateur humain est limitée. Il faut en tenir compte et réduire la *charge informationnelle et mnésique* que l'utilisateur doit mémoriser. L'ordinateur doit être considéré comme une extension de la mémoire de l'utilisateur.
La charge informationnelle signifie la quantité d'information que l'utilisateur doit mémoriser pour contribuer à l'interaction . Elle est fonction de la tâche.

R21 Concision et valeur par défaut

La charge informationnelle peut être diminuée par l'utilisation de valeurs par défauts. Deux cas sont possibles:

- valeurs par défaut statiques: elles ne sont modifiées que par l'utilisateur
- valeurs par défaut dynamiques: elles sont réévaluées par le logiciel au cours de l'interaction.

R22 Concision et macrocommandes

Le mécanisme de macro-commandes permet d'organiser les connaissances de l'utilisateur et de les regrouper dans un seul mnème.
L'utilisateur, grâce aux macroscommandes, peut exprimer directement ses intentions dans un langage de haut niveau. puisqu'elles réduisent la distance entre les intentions et la spécification du plan d'actions, elles réduisent aussi la distance entre les variables psychologiques et les variables physiques.

R23 Concision et abréviations

Les abréviations sont très utiles dans le cas d'utilisateurs experts. Elles facilitent l'exécution de ses actions: elles évitent le temps d'affichage des menus et les choix à faire.

R3 Retour d'information et guidage

Par retour d'information, on entend que l'utilisateur soit régulièrement tenu au courant de l'état du système et de l'état d'exécution de son action. Ce retour doit être immédiat et informatif. Il permet à l'utilisateur de comparer les résultats aux objectifs recherchés. Ce qui lui permet de faire aisément la correspondance entre les variables psychologiques (donc ses objectifs) et les variables physiques (celles qui définissent l'état du système. Cette correspondance est d'autant plus instantanée que la différence entre ces deux types de variables est réduite.

Le retour d'information rassure l'utilisateur sur le déroulement de sa tâche. Il permet le guidage de l'utilisateur dans son processus de décision à propos des actions à exécuter. De ce fait, certaines erreurs peuvent être évitées et le charge informationnelle est réduite.

Exemple

L'envoi des données du PC de l'utilisateur vers l'ordinateur serveur peut être présenté par un indicateur de progression pour informer l'utilisateur sur l'état du transfert.(voir fig 3.1)

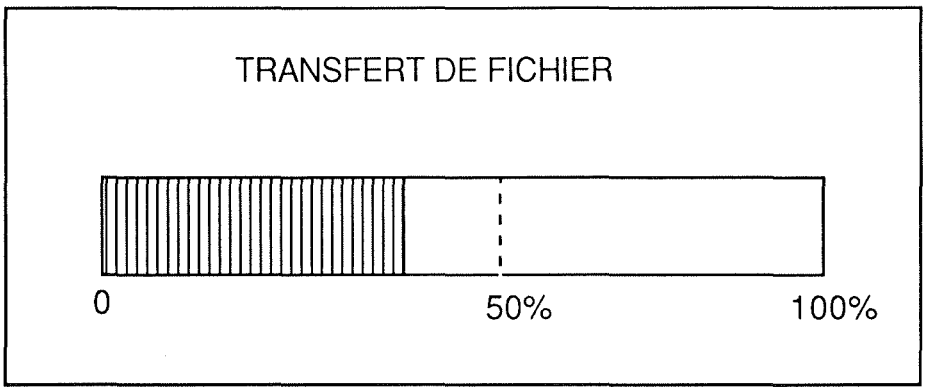


fig3.1 indicateur de progression

R4 Gestion des erreurs

L'utilisateur doit être prévenu des erreurs qu'il a commis et doit avoir la possibilité de les corriger.

R41 Signalement

Le signalement des erreurs doit être immédiat (limitation de la mémoire à court terme). Ce signalement devra apparaître dans une zone fixe commune à tous les écrans.

Exemple

Dans l'interface réalisée, une erreur est signalée par un son accompagné de l'apparition d'un message en bas de l'écran.

R42 Défaire et refaire

Toute action doit être réversible. C'est à dire l'utilisateur doit avoir la possibilité de retrouver l'état du système précédent l'exécution de son action. Ceci encourage l'utilisateur dans l'exploration des possibilités qu'offre le système et l'autorise à revenir en arrière s'il a déclenché une action qui ne correspondait pas à son intention.

Le taux erreurs peut être diminué par l'utilisation de valeurs par défaut.

R5 La flexibilité

Elle concerne l'ajustement de l'interface aux environnements et aux utilisateurs.

III.3 Recommandations pour une bonne conception

Elles se veulent des conseils pratiques, précis en faveur d'une conception conviviale d'une IHO.

III.3.1 Le dialogue

Le dialogue porte sur des saisies et des affichages. Une *saisie* est une suite finie d'entrées. Un *affichage* est une suite finie de sorties.
Le dialogue est une agrégation de saisies et d'affichage. Il concrétise la communication qui existe entre l'utilisateur et l'ordinateur.

Les propriétés générales du dialogues sont:

1) L'initiative

Le dialogue est dirigé soit par l'utilisateur, soit par l'ordinateur, soit les deux possibilités sont disponibles. Il existe donc trois niveaux de contrôle:

- interne: à l'initiative du système
- externe: à l'initiative de l'utilisateur
- mixte: tantôt interne, tantôt externe.

L'utilisateur suivant son niveau d'expérience, devrait pouvoir choisir le mode qui lui convient le mieux . Un dialogue à l'initiative de l'ordinateur peut être gênant pour un expert et un dialogue à l'initiative de l'utilisateur peut être source d'erreurs pour un non expert.

2) la flexibilité

Cette propriété regroupe deux propriétés:

- la capacité du logiciel à s'adapter aux différentes catégories des utilisateurs
- les différents moyens (procédures et options) disponibles à l'utilisateur pour la réalisation de sa tâche.

3) La charge informationnelle

Elle a été définie avant.

III.3.2 Les entrées

A. Entrée des informations

E1. Mode d'entrée

L'entrée des informations doit se faire par remplacements successifs de caractères.

E2 Changement de mode

Par exemple passer du mode clavier au mode souris ou invérsement. Les changements doivent être évités le plus possible, par exemple en assurant une équivalence fonctionnelle entre le clavier et la souris.

E3 Actions minimales

Quand des données sont dérivables à partir d'autres données déjà introduites, le logiciel doit les calculer , les afficher sans rien demander à l'utilisateur. Ceci minimise les actions de l'utilisateur.

E4 Valeurs par défaut

Elles doivent être affichées et visibles à l'utilisateur.

E5 Taille des entrées

Si les informations sont longues , alors il faut utiliser des abréviations ou découper en items.

Exemple

(1) Adresse : [ ]

(2) Rue : [ ]

B.P [ ]

Code postal : [ ]

Localité : [ ]

Au lieu de (1) il est préferable de faire (2).

E6 Rythme et ordre

Le rythme d'entré doit être conforme à celui de l'utilisateur.S'il y a des formulaires, l'ordre doit suivre celui des formulaires. Sinon, il doit suivre le déroulement logique de la tâche.

E7 Retour d'information (Feed back)

Chaque nouvelle entrée doit apparaître à l'écran sauf dans le cas de données confidentielles.

E8 Guidage

L'utilisateur peut être guidé dans son processus d'introduction de données de plusieurs manières:
- par des messages sur les données à introduire
- par des libellés de champs clairs
- par l'utilisation de codes auxiliaires: inverse vidéo, couleur, intensité lumineuse
- en indiquant si possible la longueur maximale permise
- en plaçant les unités de mesure dans les libellés des champs: éviter qu'elles soient introduites par l'utilisateur
- en éliminant toute possibilité de modification des libellés des champs par l'utilisateur

E9 Détection des erreurs

Chaque donnée doit être vérifiée et validée par le logiciel. L'utilisateur doit être averti des erreurs et doit savoir comment les corriger .

B. Dénominations [Scapin,87]

E10 Précises

La domination doit être claire et précise pour l'utilisateur même si elle ne l'est pas pour le programmeur.

Exemple:

"Classer par ordre alphabétique un fichier" est plus significatif pour l'utilisateur que " trier sur une clé d'accès".

E11 Homogènes

Une dénomination doit avoir le même sens quelque soit la transaction. Il faut éviter d'utiliser plusieurs termes par une même opération.

Exemple:
On rencontre les termes
"Quitter, terminer, sortir, finir, clôturer" pour l'action consistant à quitter le programme encours.

E12 Interruption du dialogue

A tout moment, l'utilisateur doit pouvoir interrompre le dialogue sans devoir finir la transaction engagée.

Exemple

L'utilisation de la touche d'échappement "Escape" pour interrompre le dialogue.

E13 Limitation des erreurs

L'utilisateur doit être averti s'il sélectionne une fonction invalide

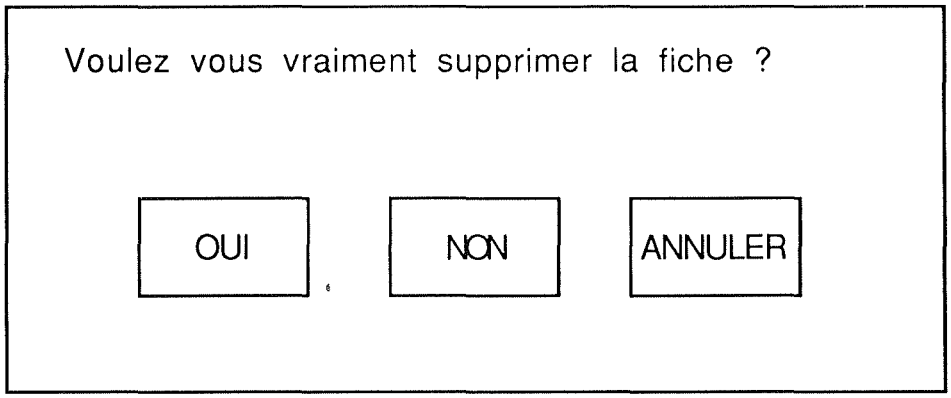
Exemple 1

La hauteur libre d'un bâtiment ne peut être modifiée par l'utilisateur.Si celui-ci sélectionne le champ correspondant à la hauteur libre, un signal sonore sera émis et un message d'erreur affiché pour signaler que la commande n'est pas valide.

Exemple 2

S'il veut effectuer une commande destructrice comme "Supprimer", une demande de confirmation doit être exprimée.

Exemple 1:



Exemple 3

Si l'utilisateur veut supprimer une unité d'isolation , une demande de confirmation sera affichée avant d'exécuter la commande.

III.3 2.Les sorties

E14 Le temps de réponse

Le temps de réponse ne doit pas être trop long, sinon l'utilisateur peut éprouvé des difficultés pour maintenir la continuité de son processus de pensée. Ce temps ne doit pas dépasser 10 à 15 secondes. L'acceptabilité du temps de réponse dépend de l'action de l'utilisateur. Un temps long pour le transfert de fichier peut être acceptable mais ne le sera pas pour une modification d'une donnée.

E15 L'affichage

A. Les ECRANS

E15.1 Visibilité

L'oeil humain doit pouvoir percevoir facilement les informations affichées. Il faut donc rendre la taille des caractères, la position, l'alignement et le contrôle adéquats.

E15.2 Lisibilité

Les éléments doivent être groupés et ordonnés de manière optimale, afin d'obtenir des groupes visuellement distincts.

E15.3 Concision

Fournir juste l'information nécessaire pour ne pas encombrer l'utilisateur.

E16 Les messages

Chaque écran doit être organisé d'une manière claire et agréable à l'oeil. Il faut éviter de le surcharger. Le titre doit être présent et séparé par au moins une ligne du reste de l'écran. Le bas de l'écran sera réservé pour les messages et les "prompts".

III.4 Conclusion

Respecter les régles ergonomiques ne garantie pas nécessairement la qualité globale de l'interface, mais montrer que l'interface ne viole pas les régles ergonomiques majeurs permet de garantir que cet interface n'est pas dépourvu de qualités de convivialité ou ne va pas à l'encontre de la manière dont l'utilisateur accomplit sa tâche.
Evidement, "les guides ergonomiques ne remplacent pas le besoin d'une bonne analyse de la tâche permettant de caractériser des spécifications et contraintes précises pour l'application future" voir référence [Bodart,91].

PARTIE III : L'INTERACTION HOMME-MACHINE

Cette partie présente et définit les différents objets qui interviennent dans l'interaction entre l'homme et l'ordinateur.
Le chapitre IV décrit les moyens d'interaction et les objets interactifs disponibles pour une interaction.
Le chapitre V présente quelques styles d'interactions pertinents pour l'interface réalisée.

Chap IV: Moyens d'Interaction et Objets Interactifs

IV.1 Les Moyens d'Interaction

IV.1.1 Définition

Un moyen d'interaction est "un dispositif à l'aide duquel l'opérateur introduit de l'information ou la fait afficher". [Provot,90]
Il existe deux catégories de moyens d'interaction:

- direct: l'introduction des informations ou des commandes se fait de manière directe sur le poste de l'utilisateur
- indirecte : sinon

A chaque dispositif est associé un ensemble d'actions permises qui engendrent certains effets.

IV.1.2 Exemples de moyens d'interaction

Une liste complète des moyens d'interaction et de leurs description est fournie dans [I.Provot,90]

Deux moyens d'interaction très utilisés que sont la *souris* et le *clavier* seront retenus ici (voir annexe A pour plus de détails).

La souris ("Mouse" en anglais) est un moyen d'interaction indirect qui sert surtout à effectuer des sélections.

Le clavier ("Keyboard" en anglais) est un moyen d'interaction direct qui sert à déplacer le *curseur* (voir annexe A) et à la saisie des informations.

Le choix entre l'utilisation du clavier ou de la souris dépend la nature de la tâche et des opérations à effectuer. Un choix minutieux peut contribuer à diminuer la complexité de la tâche comme on a vu dans le modèle de la tâche.

IV.2 Les Objets Interactifs

IV.2.1 Définition

"Un objet interactif est un objet de dialogue utilisé pour la saisie ou l'affichage d'informations relatives à la tâche de l'opérateur" [Provot,90].
Le dialogue étant une agrégation d'entrées et de sorties.

Il existe deux types d'objets interactifs: les objets composés et les objets élémentaires.

IV.2.2 Exemples d'objets interactifs

Une liste complète des objets interactifs et de leurs description peut être consultée dans [Provot,90].

Dans l'interface de notre programme, nous avons utilisé:

- le menu
- la barre de menu
- le menu déroulant
- le menu en suraffichage

CHAP V: Les styles d'interaction

Chaque application interactive utilise des moyens d'interaction et des objets interactifs pour la réalisation de saisies et d'affichage. Cet ensemble forme une *technique d'interaction.*

"Un style d'interaction constitue la combinaison logique de techniques d'interaction en vu d'organiser de manière unifiée et cohérente une suite finie de saisie/affichage" [Provot,90].

le concepteur choisit le style d'interaction qui convient en fonction de la tâche à réaliser et des utilisateurs. Un style d'interaction bien choisit contribue à faciliter l'utilisation de l'interface.

Nous nous restreignons l'étude des styles d'interactions aux quatre que nous avons employé, à savoir:

- le langage des commandes
- les systèmes de sélection des menu
- remplissage de formes
- la manipulation directe

V.1 Les Langages de Commandes

Les langages de commandes sont apparus avec les systèmes d'exploitation. Il se distinguent des langages traditionnels par deux caractéristiques:

- l'impact des commandes sur l'information est immédiat
- l'utilisateur doit mémoriser la syntaxe des commandes et déclenche lui même les actions.

Exemples de commandes

- la commande "COPY" du DOS

L'utilisateur interagit avec l'ordinateur en encodant les informations suivant la syntaxe formelle du langage

V.1.1 Fonctionnalités du système

Le succès d'un système est déterminé par les fonctionnalités qu'il offre . Ces fonctionnalités donnent à l'utilisateur le pouvoir de contrôler le système.

Déterminer le nombre de fonctionnalités que le système doit rendre disponibles est un problème délicat. Deux erreurs de conception sont souvent rencontrées:

1) un excès de fonctionnalités. Pour le concepteur cela signifie beaucoup de code à maintenir, donc plus d'erreurs à gérer, plus d'écrans d'aide et de manuels d'utilisateurs.
Pour l'utilisateur, cela se traduit par un temps d'apprentissage plus long, beaucoup de fonctionnalités à comprendre et donc de plus grandes chances d'erreurs.

2) une insuffisance de fonctionnalités L'utilisateur aura dans ce cas un sentiment de frustration parce qu' il n' a pas un contrôle suffisant du système puisque certaines fonctions utiles ou même nécessaires ne sont pas implémentées.

Une analyse approfondie de la tâche peut guider le concepteur dans le choix de fonctionnalités.

Le langage de base peut être étendu pour être utilisé à une plus large communauté d'utilisateurs par l'utilisation de *macros*. Une macro permet à plusieurs opérations d'être gérées par une seule commande. Les macros permettent d'adapter le système aux utilisateurs experts.

V.1.2 Stratégies d'organisation des commandes

V1.2.1 Liste de commandes simple

Chaque commande est associée à une sous-tâche. Il y a donc autant de commande que de tâches. La complexité du système va dés lors croître avec le nombre de tâches .
Cette stratégie est efficace si l'application nécessite un nombre très limité de tâches pour sa réalisation.

V1.2.2 Commandes avec arguments

Les commandes sont suivie par un ou plusieurs arguments qui indiquent les objets manipulés.

Exemple :

commande du DOS:
COPY FICHA FICHB pour copier le fichier FICHA dans le fichier FICHB

V.1.2.3 Commandes avec options et arguments

Les options indiquent des cas particuliers. Par exemple le nombre de pages à imprimer dans le cas de la commande d'impression.
Ces combinaisons peuvent vite devenir complexes et difficiles à mémoriser et à utiliser.

V.1.2.4 Structure de commandes hiérarchiques

Les commandes sont organisées en arbre. Le premier niveau représentera l'action de la commande, le deuxième l'objet considéré et le troisième la destination.
Cette approche a deux avantages:
- plusieurs tâches peuvent être réalisées par un nombre limité de commandes. Il suffit de combiner les commandes avec les autres arguments
- elle aide l'utilisateur non expert à maîtriser et à mémoriser les commandes.

V.1.3 Conclusion

Les langages de commandes sont orientés syntaxe et c'est de là que vient leur plus gros inconvénient: l'utilisateur doit fournir l'effort de mémorisation de la syntaxe des commandes. Cette mémorisation est d'autant plus difficile que les commandes sont complexes. Une utilisation fréquente du système peut aider l'utilisateur à comprendre le fonctionnement du système.

Un avantage de ce style d'interaction c'est qu'il peut s'adapter au travail des utilisateurs experts par la définition de macros.

Un autre avantage est que le dialogue est à l'initiative de l'utilisateur. C'est lui qui décide des actions à exécuter et c'est lui qui lance cette exécution. C'est un avantage pour les experts mais pour un débutant cela peut poser des problèmes.

V.2 Sélection de menu

Introduction et definiton

A la différence des langages de commandes, dans un système de sélection de menu, l'interaction se fait par un choix que l'utilisateur effectue dans un ensemble de commandes possibles organisées spatialement. L'utilisation ne nécessite ni apprentissage ni mémorisation de commandes.

Les menus se composent d'items. Il existe plusieurs manières d'organiser ces items et plusieurs types de sélections de menu.

V.2.1 Mécanismes de sélection

Le mécanisme de sélection est un aspect vital pour d'utilisateurs. Plusieurs mécanismes peuvent être utilisés:

- Numéroter les items
- utiliser les lettres au lieu des chiffres

Ces deux méthodes peuvent se combiner pour donner de meilleurs résultats.
Au lieu de taper leurs choix, les utilisateurs peuvent amener le curseur à l'endroit souhaité. Pour ceci, ils peuvent utiliser des dispositifs tels que la souris, les manettes les touches de tabulation ou simplement en touchant l'écran. L'item actif change alors de couleur ou d'intensité lumineuse.

V.2.2 L'organisation sémantique

La première étape de conception d'un système de sélection de menu est la création d'une organisation sémantique de la tâche qui soit claire, compréhensible et facile à mémoriser.
Organiser sémantiquement une tâche c'est la décomposer en un ensemble de catégories distinctes. Chaque catégorie regroupe un ensemble d'items liés logiquement entre eux. Un item ne devrait appartenir qu'à une et une seule catégorie. Les catégorie devraient former une partition.
Une catégorie est donc un groupe d'items. Elle constitue un menu.

Des catégories claires et faciles à comprendre donnent à l'utilisateur un sentiment de confiance dans son processus de sélection. Il sait d'avance l'effet que va engendrer tel ou tel choix.

La conception de l'organisation sémantique des systèmes de sélection de menu n'est pas une tâche simple: Le concepteur se trouve confronté à des limitations d'espace dans les écrans et à la nature de la tâche elle même.

Les applications utilisant la sélection de menu varient par le nombre de menus qui les constituent par le nombre d'items qui forment ces menus et donc par la difficulté qui en résulte.

Différentes organisations de menus sont rencontrées dans les applications:

- menu unique
- séquence linéaire de menus
- structure d'arbre stricte

- réseaux cycliques et acycliques

V.2.2.1 Menu unique

Certaines tâches ne nécessitent qu'un menu unique pour être accomplies. Les menus unique (voir fig 5.1) prennent diverses formes. Ils peuvent avoir un ou plusieurs items, nécessiter deux ou plusieurs écrans ou permettre plusieurs sélections. Ils peuvent être présents en permanence alors que l'écran principal change. Ils peuvent aussi apparaître momentanément dans la zone de travail (menus en sur-affichage).

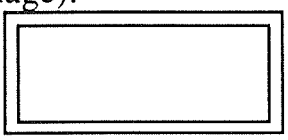


fig 5.1 menu unique

On distingue:

a) Menus binaires

C'est le cas le plus simple. Deux choix sont possibles.

Exemple

Colonne centrale OUI/NON:
Votre choix SVP:

Cette forme peut être ambiguë pour l'utilisateur débutant. Le sens exact de la requête peut lui échapper.D'autre part, la réaction du système suite à un choix négatif ou affirmatif n'est pas évidente :

- la sélection de "oui" peut entraîner une succession d'écrans que l'utilisateur ne saura contrôler
- la sélection de "non" peut à son tour entraîner n'importe quoi.

Cette incertitude sur ce qui va se passer peut susciter chez l'utilisateur un sentiment d'anxiété et de frustration.

Ces inconvénients pourront être évités en détaillant davantage les choix et en explicitant l'effet de chaque choix sur le système.

b) Menus avec plusieurs items

Les menus uniques peuvent avoir plusieurs items.

Exemple 1: Menu XTRA Astron. C'est un menu tiré de l'ancienne version du logiciel Cyprion (voir fig 5.2)

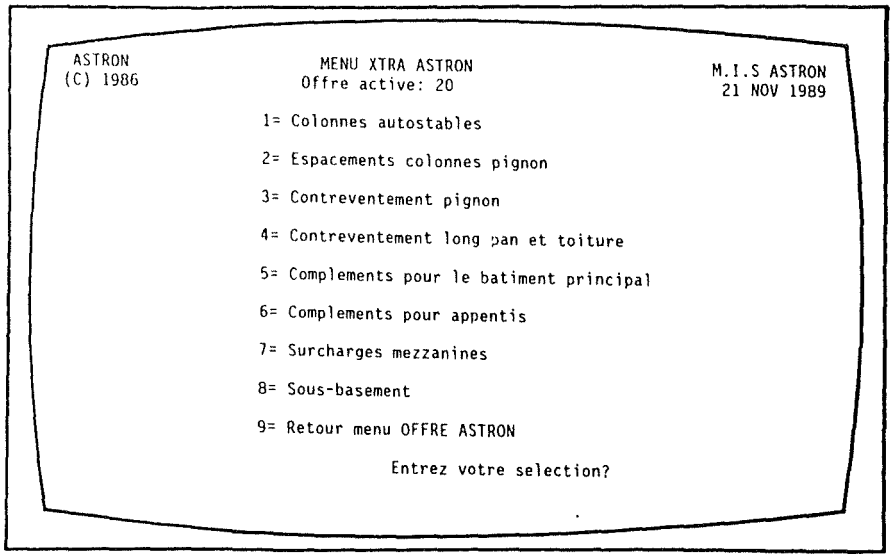


fig 5.2 Menu avec plusieurs items

Exemple 2: Menu "Fiche client " (fig 5.3)

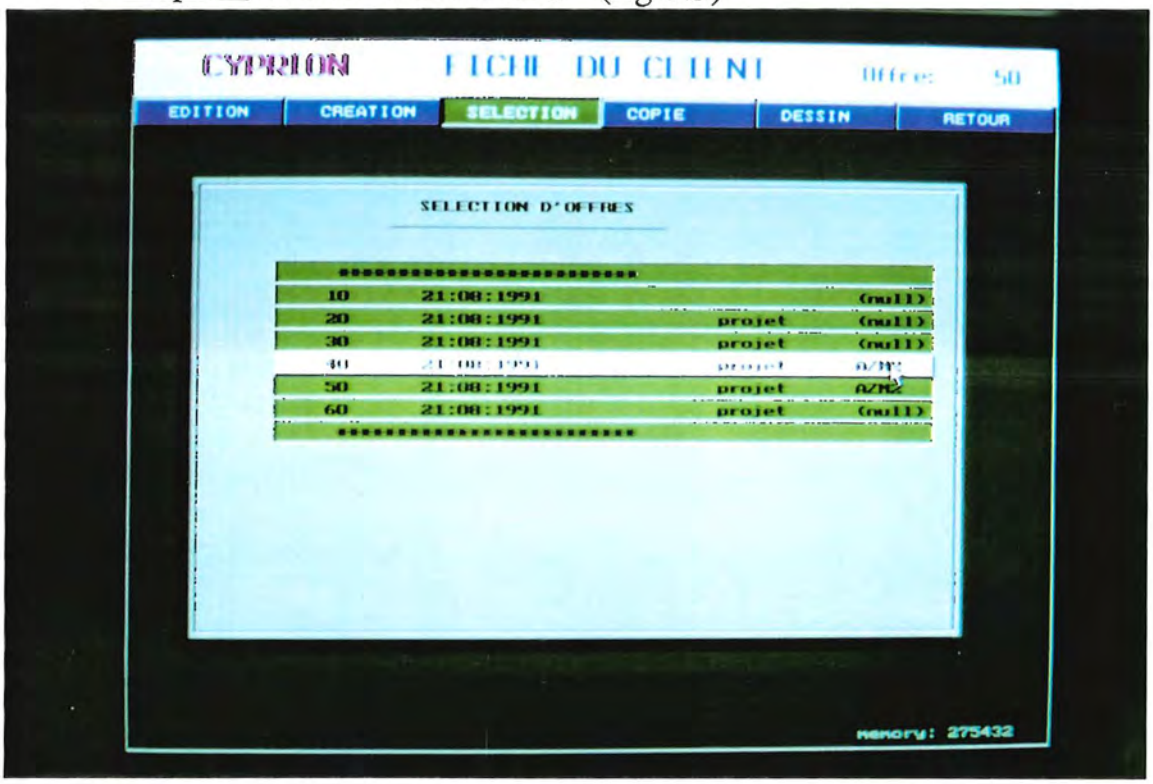


fig 5.3 Exemple d'un menu avec plusieurs items.

Exemple: "Menu Offre Astron"

fig 5.4 Exemple d'un menu déroulant

c) Menus déroulant ("pull-down" menu en anglais)

C'est un menu qui apparaît suite à l'activation d'un item du menu constituant la barre du menu.

Exemple

Le sous-menu relatif à l'introduction des données dans la version actuelle de Cyprion, apparaît suite à l'activation de l'item "Description projet" du menu principal de Cyprion
(voir fig 5.4)

d) Menus en suraffichage ("Pop-up menu" en anglais)

Ce sont des menus qui apparaissent sur la zone du travail suite à une activation d'un item par un dispositif de désignation, la souris par exemple.

Exemple: Cas du bâtiment principal (fig 5.5)

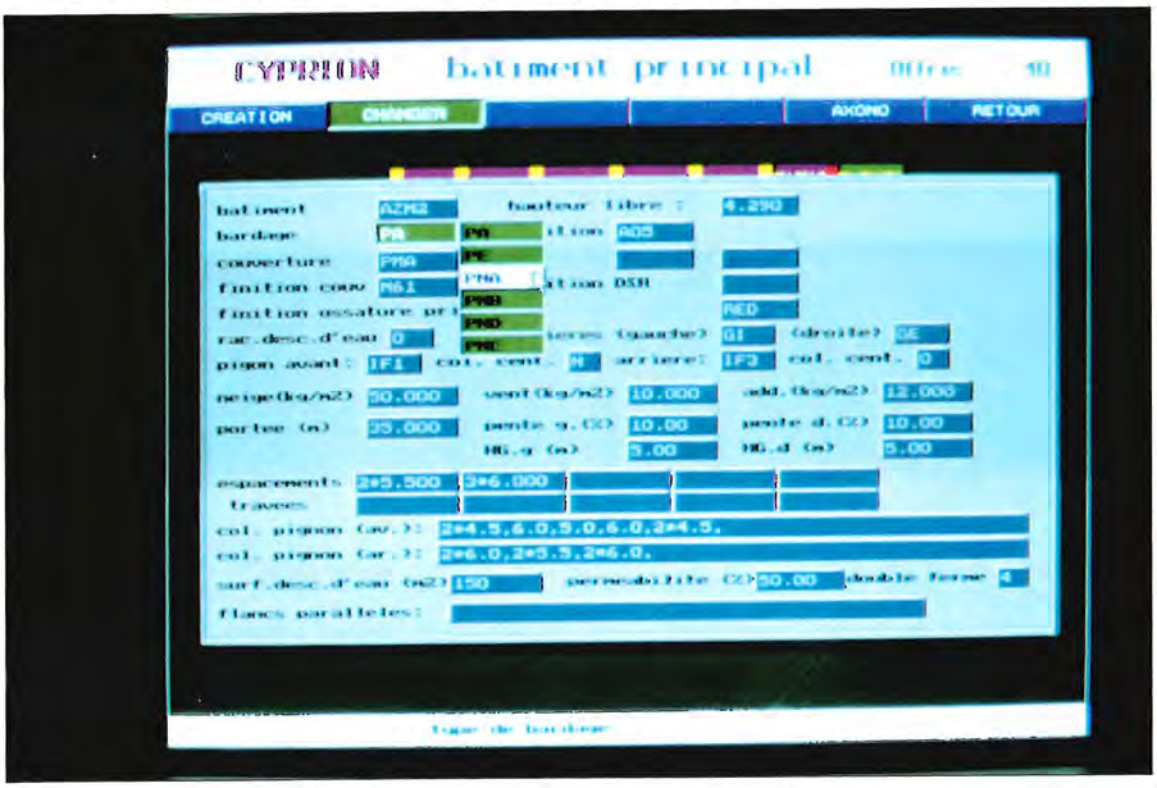


fig 5.5 exemple de menu en sur-affichage

Dans la figure 5.5, le menu constituant les différents éléments de l'introduction des données apparaît en suraffichage suite à l'activation de l'item "DESCRIPTION PROJET" dans le menu principal, par l'intermédiaire de la souris.

e) Menus permanents

Ces menus permettent d'afficher en permanence un ensemble de commandes applicables à un objet actif.

Exemple: Menu "FICHE CLIENT"

Dans la figure 5.4, les items suivants "EDITION", "CREATION", "SUPPRESSION", "COPIE" et "RETOUR" constituent les actions permises sur l'objet i"FICHE DE CLIENT". Ils restent affichés pendant tout le traitement de cet objet. Ils forment un menu permanent.

f) Menu avec sélections multiples

Offre à l'utilisateur la possibilité de faire des choix multiples parmi ceux offerts. L'utilisateur parcourt la liste avec la souris et clique à chaque fois que l'item l'intéresse pour le sélectionner.

V.2.2.2 Séquence linéaire de menus

C'est une série de menus interdépendants qui guident l'utilisateur dans une série de choix (voir fig 5.6).

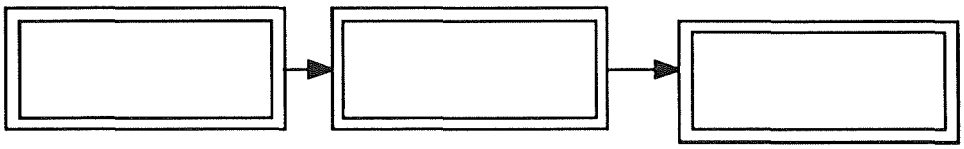


fig 5.6 structure linéaire

Avec des dispositifs de désignation et des écrans haute résolution, il est possible d'inclure dans un même écran plusieurs menus. L'interface est alors fortement simplifiée.

La séquence linéaire de menus guide et oriente l'utilisateur dans son processus de décision: elle lui présente une décision à la fois. Cette méthode peut être améliorée en affichant les choix déjà effectués. L'utilisateur sait alors quelle décisions il a prises avant et connait sa position dans la séquence de menus. Il a une idée claire sur la progression des menus et peut donc prendre facilement sa décision.

V.2.2.3 Structure d'arbre

Certaines tâches présentent un grand nombre d'items. Il est alors difficile de mémoriser l'ensemble de ces items et les liens Qui existent entre eux. Cet inconvénient peut être évité formant des groupe d'items ayant des caractéristiques commune Si l'ensemble des groupes forment une partition, on a alors une structure hiérarchique d'arbre.(voir fig 5.7).

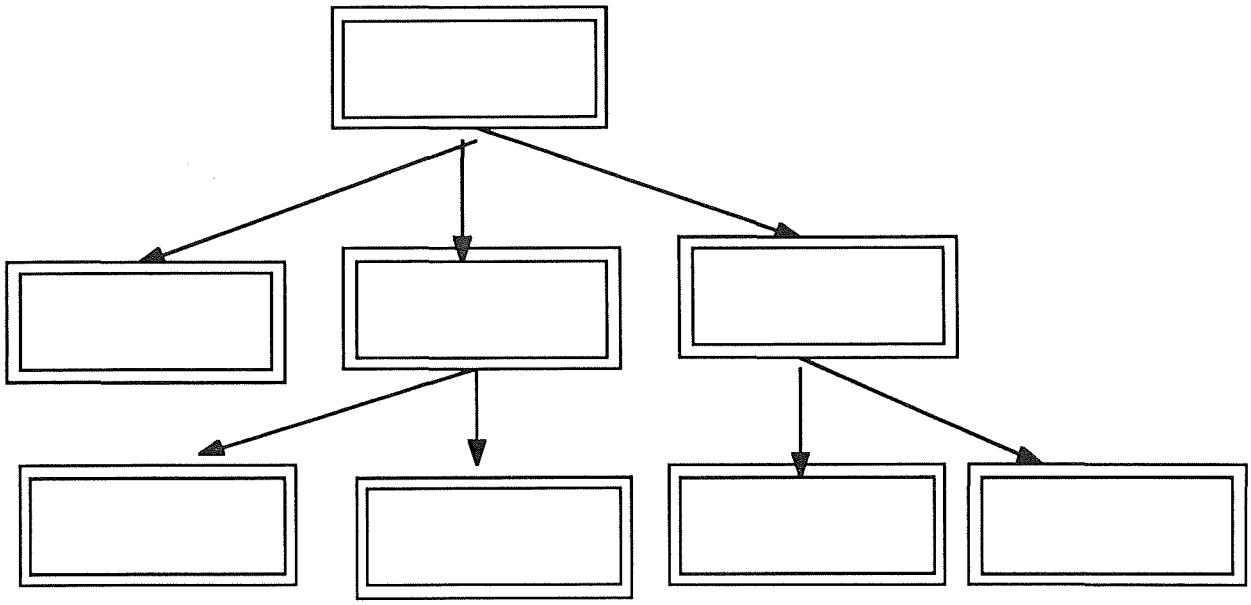


fig 5.7 structure d'arbre

Former des groupes d'items n'est pas un tâche facile. Il n'existe pas une solution parfaite unique et acceptable par tout le monde. Mais la conception initiale peut être progressivement améliorée en fonction des réactions des utilisateurs.

Les structures d'arbres sont intéressantes car, de par leur nature, elles guident l'utilisateur en structurant l'accomplissement de sa tâche.
Si les groupes s'avèrent familiers à l'utilisateur et si celui-ci sait ce qu'il cherche, alors le parcourt de l'arbre des menus est très rapide. Si par contre, l'utilisateur a une notion vague de ce qu'il cherche et si les groupe ne sont pas tés significatifs pour lui, il risque de se perdre.

Le choix d'une terminologie en relation avec la tâche de l'utilisateur s'impose car elle oriente celui-ci.

a) Profondeur et largeur d'un arbre

La *profondeur* d'un arbre est le nombre de niveaux. La l*argeur* est le nombre d'items par niveau.

Exemple

Le nombre de niveaux diminue si le plus d'items sont mis dans le menu principal C'est avantageux si cela n'entraîne pas un ralentissement dans l'affichage et donc un temps de réponse du système plus élevé. Par conséquent, déterminer le nombre de niveaux et le nombre d'items par niveau dépend de considérations ergonomiques.

Il est recommandé d'avoir ([Miller,81]):

- 3 à 4 niveaux
- 4 à 8 items par niveau.

b) groupement sémantique dans la structure d'arbre

Grouper sémantiquement les items en arbre n'est pas tâche aisée. Définir des régles générales n'est pas non plus évident.
Young and Hull (1982) [Young,82] ont cité quelques régles suite à l'étude des problèmes observés dans le système "British Pretel View Data":

1. créer des groupes d'items logiquement similaires
2. former des groupes d'items qui couvrent toutes les possibilités
3. s'assurer que les items de bas niveau ne s'associent qu'à un seul et unique item de plus haut niveau. Autrement dit, un item ne doit.avoir qu'un seul parent.
4. utiliser une terminologie familière à l'utilisateur et s'assurer que les noms des items sont distincts.

c) Carte de menus:

La difficulté de déterminer la position dans l'arbre croit avec la profondeur de l'arbre. L'utilisateur risque alors de perdre son sens d'orientation.
Pour résoudre ce problème, certains concepteurs utilisent une carte de menus. C'est un index imprimé. Il donne à l'utilisateur la possibilité de d'avoir une vue globale de l'organisation sémantique.

d) Organisation alphabétique

Trouver une organisation sémantique est souvent une tâche complexe. De ce fait, certains concepteurs ont adopté une stratégie d'indexation qui fournit une structure d'arbre basée sur une simple organisation alphabétique des items.

Aucune assertion ne permet d'affirmer laquelle des deux méthodes est la meilleure. Le choix entre les deux méthodes dépendra de la nature de la tâche à réaliser. Certains recommandent d'offrir les deux possibilités à l'utilisateur et le laisser lui même décider de celle qui lui convient.

En résumé, il n'y a pas de structure idéale qui correspond exactement au attentes de l'utilisateur. Le concepteur doit user de son savoir, de son jugement et de sa connaissance de la tâche de l'utilisateur pour concevoir une structure appropriée.
Il n' y a pas de structure idéale, univoquement déterminée mais on dispose de conseils à mettre en oeuvre pour qu'on en obtienne pas une qui soit déficiente.

V.2.2.4 Réseaux cycliques et acycliques

On a un réseau :
- acyclique : si on peut atteindre un menu par plus d'un chemin (voir fig 5.8)
- cyclique : si des menus avec des chemins différents permettent aux utilisateurs de répéter les menus (voir fig 5.9).

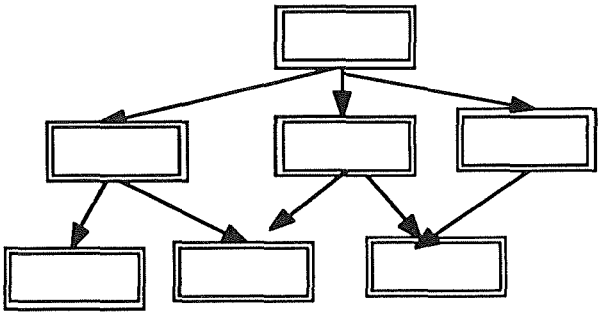


fig 5.8 Réseau acyclique

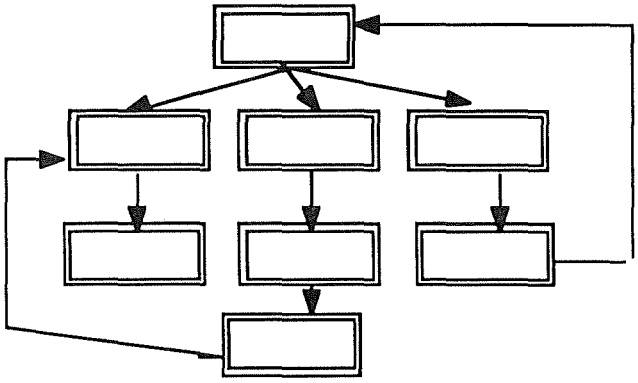


fig 5.9 Réseau cyclique

Les réseaux sont utiles dans certains cas. Par exemple, il permettent à l'utilisateur d'accéder à un menu à partir de menus différents, ou bien de retourner à un niveau supérieur pour commencer une recherche, sans avoir à tout recommencer dés le début.
La possibilité que l'utilisateur se perde est plus grande dans les réseaux. Avec une structure d'arbre, il n'y a q'un seul parent. L'utilisateur peut sans problème remonter l'arbre. Avec les réseaux, il est nécessaire de maintenir une liste des menus déjà visités pour permettre des retour en arrière si nécessaire.
Il est souhaitable de garder la notion de niveau ou de distance par rapport au menu principal pour réconforter l'utilisateur et diminuer sa désorientation.

V.2.3 Organisation et présentation des items

Ordonner des items séquentiellement constitue une difficulté. Certains items s'organisent et forment des séquences qui suivent un ordre naturel tel que:
- ordre chronologique
  Exemple: janvier, février, mars...
- ordre numérique
  Exemple: 10, 20, 30...
- propriétés physiques: longueur, surface, volume, température, poids, vitesse...

Exemple: soient les items: bouteille, tasse, verre de volumes respectifs : 10, 6, 9. Ils peuvent être ordonnés selon leur volume. On a alors la séquence suivante: tasse, verre, bouteille.

Quand il n'y a pas d'ordre naturel, le concepteur peut choisir entre les possibilités suivantes:

- séquence alphabétique des termes
- groupement d'items en relation entre eux et séparés par des séparateurs tels que les blancs
- d'abord les items les plus utilisés
- d'abord les items les plus importants: l'importance est une notion difficile à définir car variant d'un utilisateur à l'autre.

V.2.4 Temps de réponse et taux d'affichage

Le temps de réponse est le temps nécessaire au système pour qu'il commence à afficher de l'information en réponse à la sélection de l'utilisateur.

Le taux d'affichage est le nombre de caractères affichés par seconde. Ces deux variables sont liées par la relation suivante:

La vitesse de parcours est un paramètre important dans l'acceptabilité d'un système.

Si l'arbre est profond et les parcours complexes et si le temps de réponse est élevé, l'utilisateur peut être ennuyé par les multiples attentes qui en résultent. Si d'un autre côté, le taux d'affichage est lent car les menus sont plus longs, l'utilisateur doit de nouveau attendre. C'est une situation de compromis que le concepteur doit essayer de résoudre.

V.2.5 Parcours rapide de l'arbre

Même si le temps de réponse et le taux d'affichage sont optimaux, devoir parcourir plusieurs menus avant d'exécuter une tâche peut contrarier un utilisateur fréquent ou expert. Il serait souhaitable de permettre des *raccourcis* ("shortcuts" en anglais).

Il existe trois approches pour permettre un parcours rapide des menus:

- l'approche "typeahead"
- l'accès direct
- l'utilisation de macros.

a) **L'approche "typeahead"**

Elle permet des "typeahead". C'est à dire que l'utilisateur peut faire ses sélections sans attendre l'affichage des menus intermédiaires. Il peut taper ses choix sous forme d'une liste de caractères lorsqu'il est en présence du menu principal.
Si les items sont identifiés par des lettres uniques, la concaténation des sélections des différents menus génère le nom d'une commande qui acquiert une valeur mnémonique. Ces mnémoniques sont mémorisés comme un seul concept. On remarque que l'on approche ici des langages de commandes où on peut utiliser des commandes groupées.

Exemple 1

Considérons l'ancienne version de Cyprion.
Supposons que l'utilisateur veut introduire des sous-basements. Au lieu de parcourir tous les menus successivement, (fig 5.10 (a), 5.10 (b) et 5.10 (c)), il peut taper les choix sous forme d'une liste de caractères. Dans ce cas, la liste "2118".

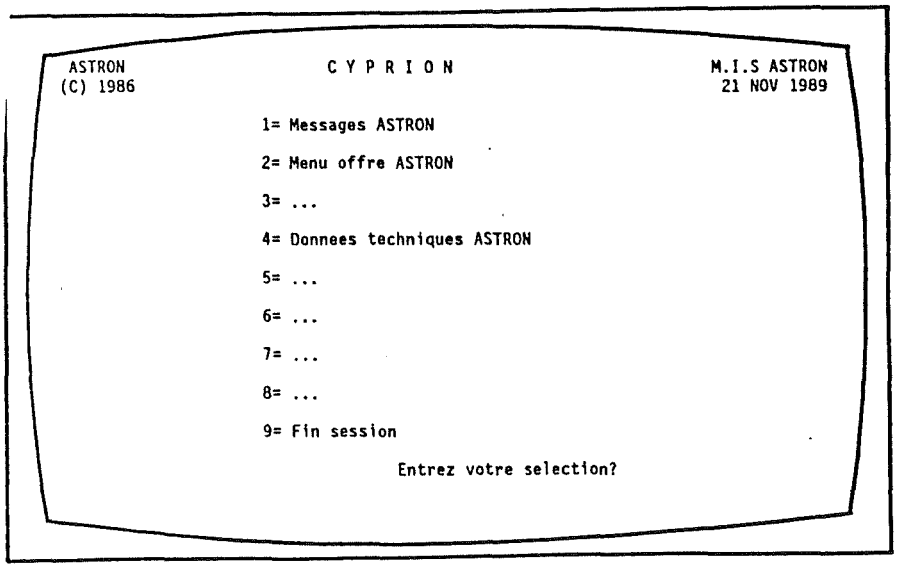


fig 5.10 (a) premier choix du premier menu

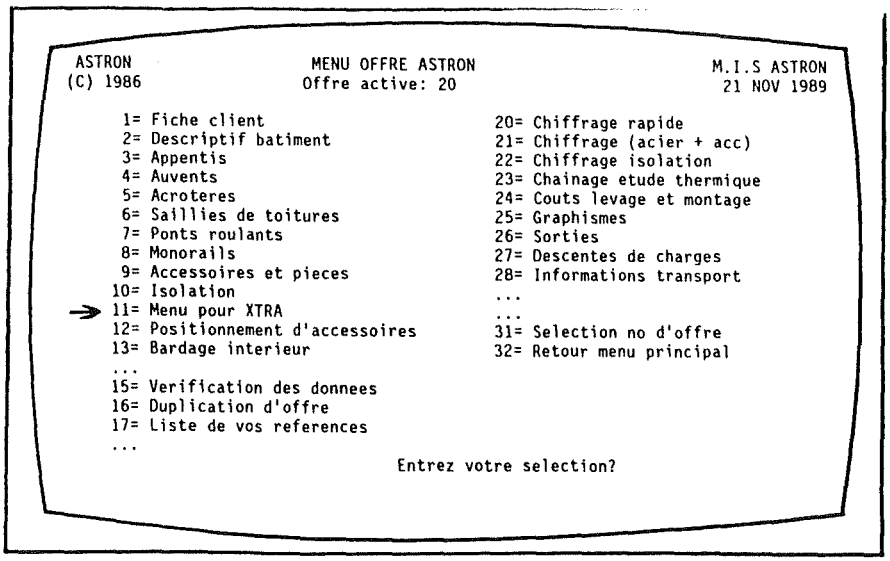


fig 5.10 (b) deuxième choix

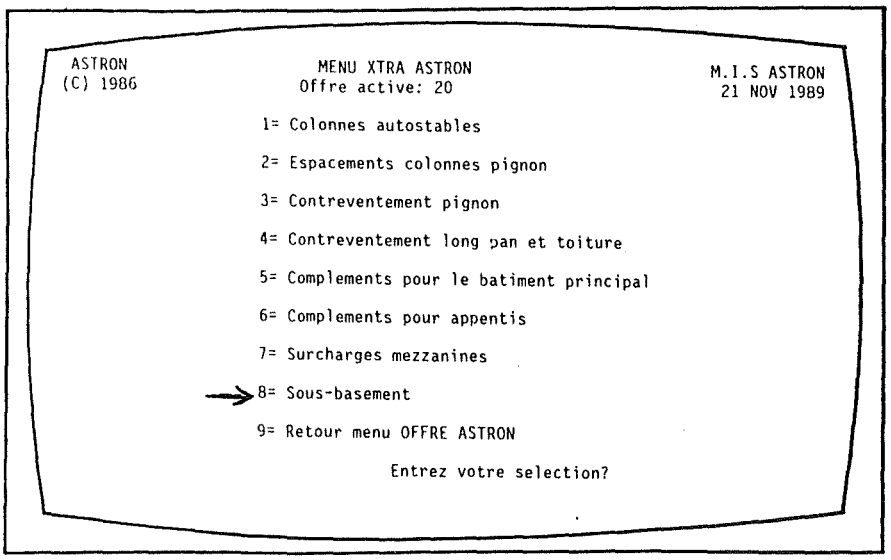


fig 5.10 (c) troisième choix

b) **L'approche "accès direct"**

Les items des menus sont numérotés et chaque menu est identifié par un nom. Il suffit aux utilisateurs de taper le nom du menu pour y accéder directement.
Cette stratégie est efficace si le nombre de menus que l'utilisateur doit mémoriser n'est pas élevé. Il est nécessaire de tenir à jour une liste des noms des menus pour qu'un même nom ne soit pas affecté à des menus différents.i.e pour préserver le caractère identifiant des noms des menus.

Exemple

Considérons toujours le cas de l'exemple 1. On suppose qu'à chaque menu a été attribué un nom. Si le menu "Sous-basements" est désigné par "SB", l'utilisateur n'aura dans ce cas qu'à taper ce nom. Il accédera alors directement au menu des "Sous-basements".

c) **L'approche "macros de menu"**

L'objectif est de permettre aux utilisateurs d'enregistrer des chemins fréquemment utilisés et régulièrement empruntés sous forme de macros de menu. Autrement dit, leur permettre de définir leurs propres commandes.
A l'appel d'une macro, l'exécution du contenu de la macro se fait automatiquement. Ce mécanisme permet d'adapter le système aux utilisateurs. Il fournit un mécanisme d'accès simple aux utilisateurs qui ont des besoins limités.

Exemple 3

Considérons à nouveau l'exemple 1.
Ici, c'est l'utilisateur définit lui-même une macro.qui correspond au chemin emprunté. Soit "MO-XT-SB" l dans notre exemple. Il lui suffit alors de taper le nom de cette macro pour accéder au menu des "Sous-basements".

V.3 Remplissage de formes

Pour certaines tâches, il y a beaucoup de données à introduire, des noms, des valeurs numériques... Dans ces cas, l'utilisation du clavier devient. préférable. Celui-ci peut être vu comme un menu unique continu où plusieurs sélections sont possibles.

Le remplissage de formes est utilisé quand il y a plusieurs champs de données. Cette approche est attirante car toutes les informations sont visibles, ce qui donne à l'utilisateur un sentiment de contrôle du dialogue. La représentation

de cette forme peut ressembler à un formulaire habituel imprimé. (voir fig 5.11)

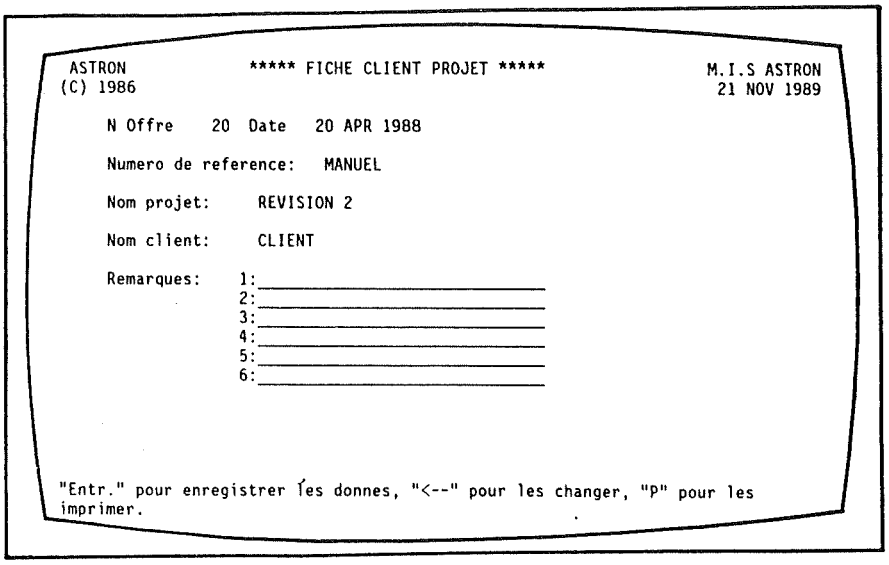


fig 5.11 Exemple de remplissage de forme

Le remplissage de formes est utilisé quand il y a plusieurs champs de données. Cette approche est attirante car toutes les informations sont visibles, ce qui donne à l'utilisateur un sentiment de contrôle du dialogue. La représentation de cette forme peut ressembler à un formulaire habituel imprimé. (voir fig 5.12)

V.3.1 Règles pour une bonne conception

Ces règles sont basées sur des cas pratiques et sur des observations faites sur des systèmes déjà réalisés [Sneiderman,87]:

1. titre significatif: le titre doit identifier le sujet
2. instructions compréhensibles: les instructions doivent être brèves et claires. Les tâches doivent être décrites dans une terminologie familière à l'utilisateur.
3. groupes logiques et séquencement des champs: regrouper les champs liés logiquement. Séparer ces groupes par des espaces blancs. Ordonner les données communes par exemple: ville...état...code...
4. présentation sous forme attirante: avoir une distribution de champs uniforme, des alignements symétriques. Ceci donne un sens d'ordre et de compréhension à l'utilisateur.
5. étiquettes de champs familières: utiliser des termes familiers et communément admis.

6. terminologie et abréviations cohérentes: utiliser une même abréviations pour le même terme. Former une liste d'abréviations pour garder un tout cohérent.
7. limites et espaces visibles pour entrer les données: afin que l'utilisateur sache le nombre de caractères à introduire.
8. mouvement du curseur approprié: mécanisme doit être simple et visible.
9. correction des erreurs pour les caractères individuels et les champs en entier: possibilité de retourner en arrière pour des corrections.
10. messages d'erreurs pour des valeurs non acceptables: le message doit indiquer les valeurs permises
11. champs optionnels doivent être marqués: de préférence, les afficher après les champs non optionnels.
12. messages d'explication des champs: donner quand c'est possibles des informations sur le champs actif ou ses valeurs.permises Ces informations apparaîtront dans une position standard
13. signal de fin: pour indiquer aux utilisateurs ce qu'ils doivent faire quand ils ont fini d'introduire les données.

Ces régles peuvent paraître simples et évidentes, pourtant elles ne sont pas toujours respectées.

Résumé

Le concepteur doit avant tout analyser la structure sémantique de l'application. Il doit choisir l'organisation la plus appropriée à la tâche de l'utilisateur. Si l'utilisateur est expert ou utilise fréquemment le système, des raccourcis et des macros de menus doivent être possibles.

v.4 Manipulation Directe

Pour mieux comprendre ce qu'est la manipulation directe, voyons quelques exemples:

V.4.1 Exemples

Exemple 1 (fig 5.12)

Considérons une tâche de l'application relative à l'introduction de données d'une offre à chiffrer. Soit la tâche qui consiste à traiter l'isolation. L'utilisateur, dans un premier temps a isolé trois parties du pignon avant . Il se rend compte dans un deuxième temps que l'isolation des trois parties va coûter cher et decide de n'isoler que la partie destinée aux bureaux. Il aimerait retirer l'isolation d'une partie de bâtiment. Avec la souris, il se positionne sur l'unité d'isolation qu'il veut annuler et la sélectionne. Cette partie se colore en rouge pour indiquer le danger de destruction de l'unité. La couleur rouge a été choisie car en général elle est synonyme de danger potentiel chez les gens. Ceci permet à l'utilisateur d'annuler la commande de suppression s'il change d'avis. S'il confirme la suppression, le résultat est immédiat. Il voit sur l'écran la disparition de l'unité d'isolation sélectionnée.

Exemple 2: (fig 5.13)

Considérons le cas de l'introduction des colonnes autostables. L'utilisateur a devant lui le graphique des bâtiments avec les distributions des colonnes. Avec la souris, il contourne toutes les colonnes qu'il veut rendre autostables.Le déplacement de la souris engendre alors une ligne continue verte qui délimite les colonnes sélectionnée. Quand l'utilisateur a fini la sélection, il lâche le bouton de la souris La ligne se ferme alors sur les colonnes sélectionnées et celles-ci deviennent immédiatement rouges Le rouge ici n'a rien avoir avec le danger, mais c'est seulement pour bien distinguer entre les colonnes autostables et les colonnes normales.

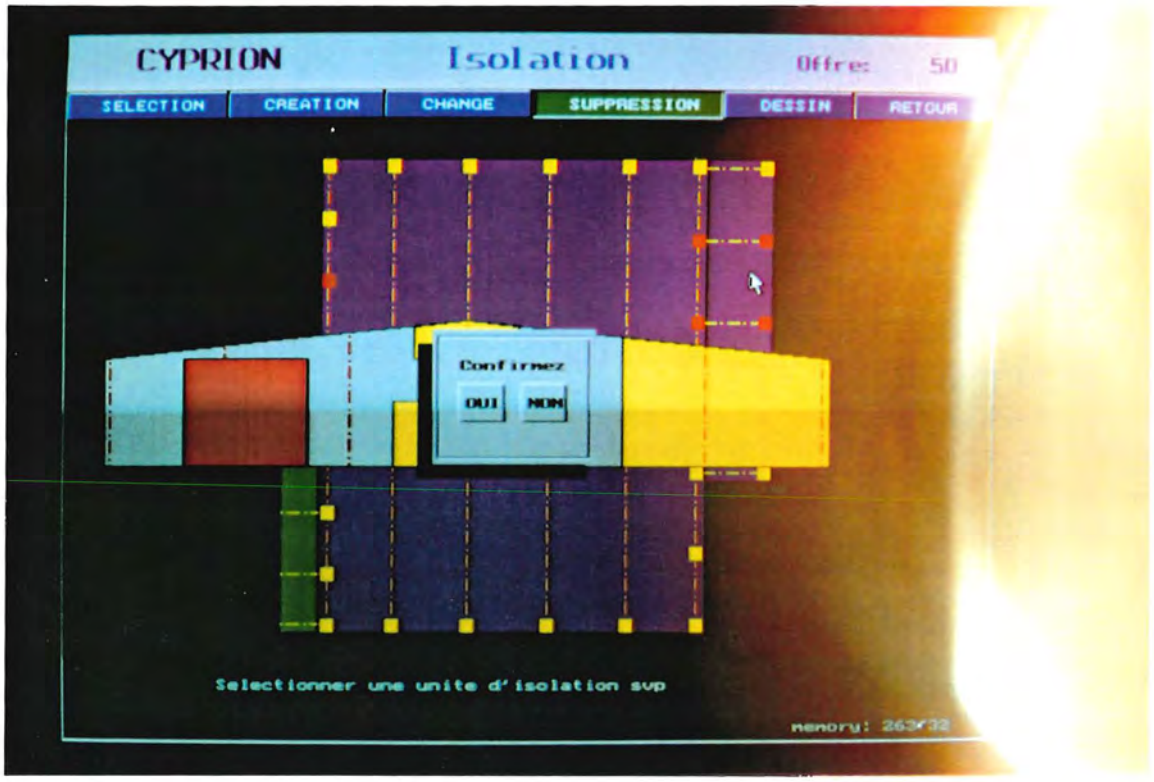


fig 5.12 Selection d'une unité d'isolation

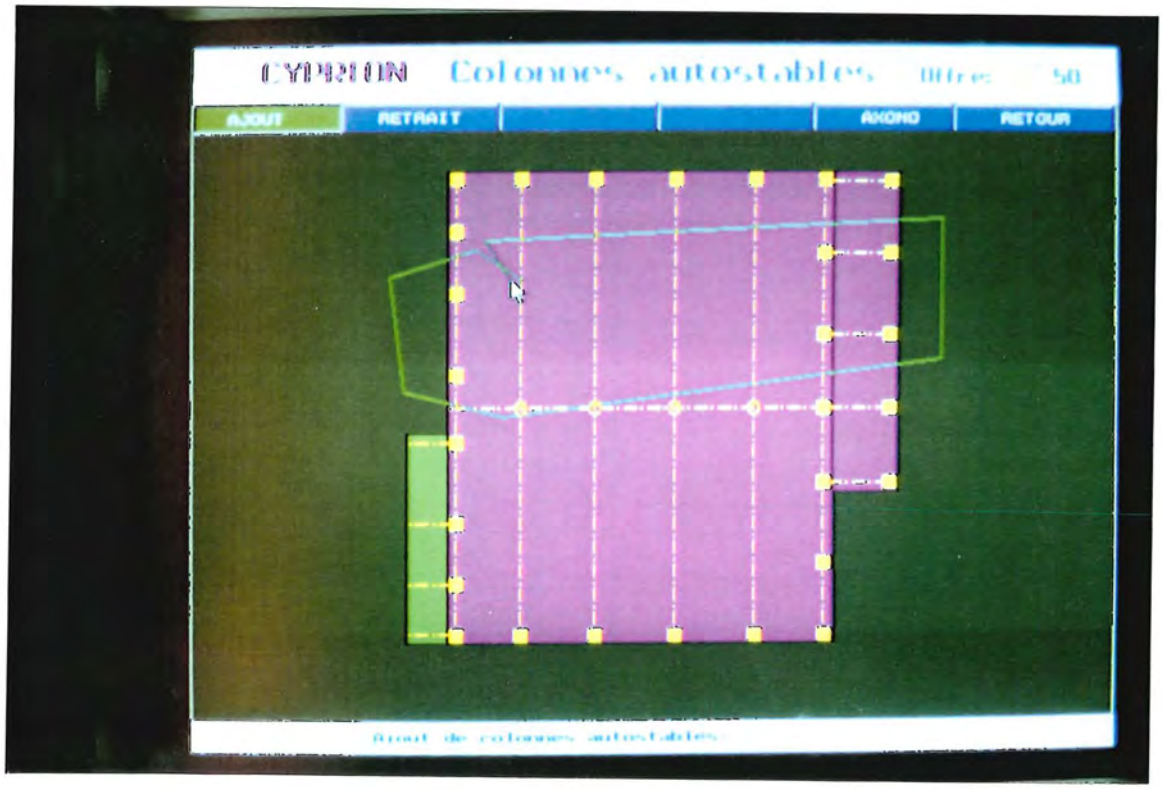


fig 5.13 Sélection de colonnes autostables

En fait, l'idée centrale derrière la manipulation directe est la visibilité des objets et des actions intéressantes, des actions rapides et réversibles.

Les utilisateurs sont contents d'utiliser ce genre de systèmes. Ils ont l'impression de maîtriser le système, d'être plus performants dans la réalisation de leurs tâches. Ils sont plus confiants et sont désireux d'explorer plus en profondeur le système.

V.4.2 Explication de la manipulation directe

Shneiderman [Shneiderman,87] définit les systèmes à manipulation directe comme ceux ayant les propriétés suivantes:

1. représentation continue des objets et actions intéressantes
2. actions physiques ou boutons à enfoncer au lieu d'une syntaxe complexe
3. opérations rapides, incrémentales dont l'impact sur l'objet est immédiatement visible.

Le psychologue Jean Piaget cité par Shneiderman [Shneiderman,87] parle de quatre étape dans l'évolution de l'individu:

1. sensorimoteur: se situe de la naissance à l'âge de deux ans.
2. préopérationnel : de deux ans à sept ans
3. opérationnel concret: des sept ans à onze ans
4. opérations formelles: à partir de onze ans.

Les actions physiques sur un objet sont compréhensibles pendant l'opérationnel concret. L'étape des opérations formelles est l'étape de la manipulation de symboles pour représenter des actions sur des objets. A cette dernière étape, il faut fournir un plus grand effort mental pour pouvoir lier la représentation symbolique à l'objet considéré. Le but de la manipulation directe est justement de ramener l'activité à l'étape de l'opérationnel concret pour faciliter les tâches. En général, les représentations et actions spatiales sont plus significatives et plus faciles à comprendre par l'être humain. Les représentations linguistiques sont moins évidentes.

V.4.3 Limites de la manipulation directe

La complexité de programmation des interfaces utilisant la manipulation directe croit très vite avec la complexité de l'application. Le concepteur doit trouver une représentation appropriée pour organiser et visualiser les différents objets de l'application. Il doit apprendre à penser dans une forme visuelle.

En fait, les limites de la manipulation directe dépendent de l'imagination et de l'habileté des concepteurs.

V.4.4 Conclusion

Les systèmes à manipulation directe offrent en général une interface utilisateur conviviale et facile à utiliser.
Mais on a souvent tendance à oublier le fait que la facilité de l'utilisation d'un tel système croit avec la complexité de sa programmation

V.5 Conclusion

Comme nous avons utilisé conjointement quatre styles d'interaction, il paraît intéressant de comparer les différents styles pour voir leurs avantages et leurs inconvénients.

Les langages de commandes:

Avantages:
- flexibilité: adaptables à des environnement variés et à une large communauté d'utilisateurs
- possibilité de définir des macros
- les actions à l'initiative des utilisateurs

Inconvénients:
- mémorisation de la syntaxe des commandes
- gestion restreinte des erreurs
- nombre élevé de frappes

Les langages de commandes sont recommandés pour les utilisateurs experts et qui utilisent fréquemment le système.

Sélection de menu:

Avantages:
_ peu d'apprentissage
- nombre minime de frappes
- guide l'utilisateur dans son processus de décision

inconvénients:
- problème d'espace à l'écran
- exécution lente s'il y a beaucoup de menus

La sélection de menu se veut un style d'interaction approprié aux débutants. L'utilisation de raccourcis, l'optimisation du taux d'affichage et du temps d'exécution peuvent rendre ce style attrayant même pour les experts.

Remplissage de formes:

Avantages:
- simplification de la saisie des données
- ne nécessite pas un apprentissage approfondi
- une bonne gestion des erreurs

inconvénients:
- problème d'espace à l'écran

Le remplissage de forme est approprié aux débutants.et aux utilisateurs intermédiaires. Mais si les dispositifs d'affichages sont rapides, ce style d'interaction peut devenir intéressant pour les experts également.

Manipulation directe

Avantages:
- visibilité des objets manipulés
- apprentissage facile
- rétention aisée
- possibilité d'éviter les erreurs
- encourage l'exploration des fonctionnalités du système
- grande satisfaction subjective

inconvénients:
- programmation complexe

Ce style d'interaction est approprié aux non experts.

Il est difficile de tirer une conclusion de cette comparaison et de dire lequel des systèmes est le meilleur. Chacun des système a ses avantages et ses inconvénients. L'acceptabilité d'un système dépend surtout de ses utilisateurs. Un expert sera content avec des langages de commandes alors qu'un novice sera perdu mais sera plus assuré avec un système de sélection de menu.C'est pourquoi nous avons voulu combiner les différents styles d'interaction pour rencontrer des classes d'utilisateurs variées.

D'un autre côté, on a vu que des mécanismes peuvent être utilisés pour accélérer l'exécution des actions. Ces mécanismes tendent à rapprocher les frontières entre les différents styles d'interaction.

PARTIE 3: ANALYSE DE LA TACHE ET EVALUATION DE L'INTERFACE

chap VI: Evaluation ergonomique de l'interface

Introduction

L'analyse de Cyprion se fera en plusieurs étapes.
- Une analyse de la tâche et des utilisateurs, en complément de la description générale présentée au chapitre1
- une analyse fonctionnelle de l'interface identifiant les objets et les actions de l'interface
- étude ergonomique vérifiant *a posteriori* le respect des critères et régles ergonomiques. Cette analyse sera complétée par une mesure de la convivialité de l'interface.

VI.1 Analyse de la tâche et des utilisateurs

Le but de cette analyse est d'identifier les différents objets et concepts qui définissent la tâche de l'utilisateur, ainsi que les exigences et contraintes qui sont liées à ces objets. L'objectif est de bien cerner les caractéristiques de la tâche et les séquencements logiques des actions.

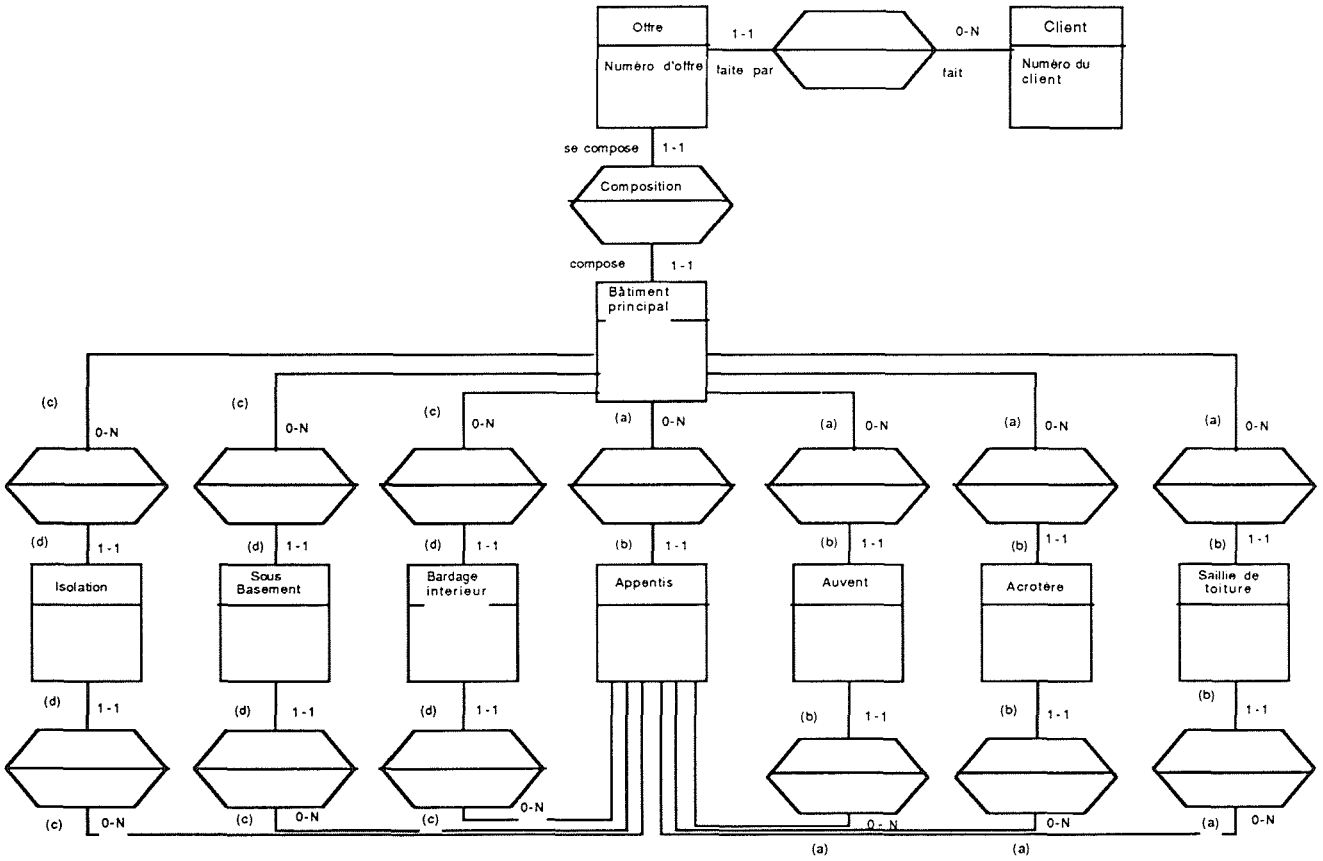
Identifier la population des utilisateurs est aussi important pour définir les fonctionnalités que l'interface devra fournir pour vérifier disponibilité.

VI.1.1 Objets de la tâche

La tâche considérée est l'introduction des données qui définissent un bâtiment industriel à chiffrer.

Les objets qui interviennent dans la réalisation de la tâche sont ceux décrits dans le chapitre. Ils sont:
- le bâtiment principal
- les appentis
- les auvents
- les acrotères
- les saillies de toitures
- les sous-basements

A ces objets, sont reliés ses concepts qui deviennent à leur tour des objets et qui sont:
- l'isolation
- la bardage intérieur



(a) se compose-1
(b) compose-1
(c) se compose-2
(d) compose-2

Fig 6.1 Schéma entité/association d'une offre

La figure 6.1 illustre le schéma entité/association [Bodart,89] qui relient ces différents objets entre eux ainsi que les contraintes auxquelles ils sont soumis.

La tâche de l'utilisateur consiste à introduire les données qui constituent les variables et paramètres définissant ces objets.

Organiser et regrouper ces données dont a besoin l'utilisateur relève du domaine de l'étude d'opportunité et de la conception, phases que nous considérons ici déjà réalisées.

VI.1.2 Organisation des objets

L'organisation des objets importe beaucoup pour l'interface considérée. C'est un des objectifs de la refonte de la partie d'introduction des données.

En effet, les objets ont été regroupés différemment pour faciliter l'apprentissage, la compréhension et le travail de l'utilisateur.

En particulier, toutes les données associées à un même objet ont été regroupés sous un seul concept. Ce qui n'était pas le cas dans l'ancienne version.

On retrouve donc les concepts:

- bâtiment principal: il englobe toutes les données qui sont de loin ou de près associées au bâtiment principal. Dans l'ancienne version, ces données se trouvaient dans deux groupes différents:
  * description du bâtiment
  * compléments pour bâtiment principal

- architecturaux: ils regroupent les appentis, les auvents, les acrotères et les saillies de toitures. Ces bâtiments sont logiquement liés entre eux. car ce sont des annexes qui peuvent être accoler au bâtiment principal ou aux appentis. Dans l'ancienne version, ces bâtiments étaient considérés comme des objets distincts et n'étaient pas regroupés.

- isolation: c'est un concept à part qui peut être associé aux objets. Dans l'ancienne version, un bâtiment ne pouvait pas être isolé partiellement. Dans la version actuelle, l'isolation partielle est disponible.

- bardage intérieur: c'est une fonctionnalité nouvelle offerte par l'interface.

- sous-basements: de même que l'isolation et le bardage intérieur, les sous-basements peuvent être placés sur les murs, en entier ou partiellement.

Tous ces objets déterminent une **offre**. Chaque offre est faite par un client particulier. La société a plusieurs clients à répertorier ainsi que les offres qui leurs sont associées. Un objet supplémentaire intitulé "Fiche client" permet de regrouper ces données et d'identifier le client et son offre.

VI.1.3 Opérations permises sur les objets

L'utilisateur doit pouvoir manipuler ces objets et effectuer des actions et opérations pour réaliser ses objectifs. Sur chaque objet sont définies les opérations permises.

Fiche client/offre:

- création de la fiche client/offre
- modification de la fiche client/offre associée à l'offre active
- sélection: activer une offre existante
- copie : duplication d'une offre. La nouvelle offre recoit un numéro d'offre identifiant, attribué par le logiciel.
- dessin : représenter graphiquement l'offre active pour autant que le bâtiment principal soit défini, l'offre active

Bâtiment principal

- création: introduction des données
- modification: modifier les données introduites

Isolation / bardage intérieur / sous-basements

- création
- modification
- suppression
- dessin

L'action "dessin" permet de représenter graphiquement les bâtiments vus de haut.

Architecturaux

- création
- modification
- suppression

VI.1.4 Les utilisateurs

Il est impératif pour le concepteur de connaître les utilisateurs potentiels afin de définir les caractéristiques de l'IHO, comme le choix des styles d'interaction.

En général, les utilisateurs sont classés en niveau:

- débutant
- novice
- intermédiaire
- expert
- maître

Les utilisateurs de l'interface de Cyprion sont les concessionnaires de la société. Ils sont supposés bien connaître la tâche. On peut distinguer deux groupes parmi les concessionnaires:

- le groupe des "anciens" qui ont déjà travaillé avec l'ancienne version de l'interface
- les "nouveaux" qui n'ont jamais utilisé l'ancienne version.

Les premiers sont familiarisés avec certaines fonctionnalités et une certaine façon de modéliser les choses. Ils auront plus de faciliter à s'adapter et à maîtriser le nouveau système.

La connaissance de la tâche n'implique pas nécessairement la facilité que peut éprouver l'utilisateur, mais elle y contribue. En plus, une nouvelle structure et une nouvelle organisation des données ont été décidées dans le but d'offrir de nouvelles fonctionnalités telles que le bardage intérieur.

D'un autre côté, les connaissances des utilisateurs et la maîtrise du système évoluent avec le temps.

Les utilisateurs diffèrent aussi par la fréquence d'utilisation de l'interface: certains concessionnaires distribuent plus de bâtiments que d'autres. Cela dépend de plusieurs facteurs dont le lieu géographique.

Ayant défini les utilisateurs, on peut caractérisé la tâche:

- elle est en générale aisée
- elle est souvent répétitive
- pour certains utilisateurs, elle est fréquente, pour d'autre, elle est rare
- elle peut être créative dans une certaine limite, mais le plus souvent, elle est routinière

VI.2. Analyse fonctionnelle de l'interface

VI.2.1 Les moyens d'interaction utilisés

Deux moyens d'interaction sont mis en oeuvre dans l'interface:
- le clavier pour l'introduction de données alphanumériques
- la souris pour la sélection de menus, de données ou de graphiques.

Le passage du mode souris au mode clavier et l'inverse se fait automatiquement selon le type de menu et selon le type de données à introduire ou à afficher.

VI.2.2 Les objets interactifs

Les objets interactifs utilisés sont:
- la barre de menu
- le menu déroulant
- l'item de menu
- le libellé
- le champs d'édition multilinéaire
- la liste de sélection
- la liste de sélection déroulante
- l'l'indicateur de progression

VI.2.3 Les types d'interaction

Les styles d'interaction présents dans l'interface sont:

A. Sélection de menu

Les choix possibles sont regroupés dans des menus. Un menu est associé à une sous-tâche. L'écran de chaque menu respecte une certaine présentation et une certaine sémantique cohérente.

A.1 Présentation

Chaque menu se compose d'une barre de menu qui contient les options représentant les opérations possibles sur les objets du menu. La liste de ces objets est présentée sous la forme d'un menu déroulant

Les items de la barre des menus représentent les types d'actions possibles. Les items des menus déroulants représentent les objets sur lesquels sont effectuées les actions. Ce choix provient de plusieurs faits:

- au niveau du menu en question, la liste des actions possibles est la même pour tous les objets.
- au niveau de l'ensemble des menus, on retrouve à une action près, le même ensemble des actions possibles.
- ce choix permet d'automatiser les procédures d'affichage des menus
- il facilite le travail de l'utilisateur: car il a un objet sur lequel il peut effectuer un ensemble d'actions. Si l'autre option a été choisie, l'utilisateur sera en présence de plusieurs objets à la fois et devra chercher les actions pour les exécuter. Ceci risque de l'embrouiller.

Chaque menu possède un titre significatif et unique.

La sélection d'un item du menu se fait par l'intermédiaire de la souris. L'utilisateur positionne la souris sur l'item et clique dessus.

A.3 Organisation sémantique

Les items d'un menu appartiennent à une même catégorie sémantique. Ces catégories sont disjointes, dérivables de l'analyse de la tâche réalisée.

A.3 Graphe de transition des menus

Il faut noter que l'on ne considère ici que la tâche consistant à introduire et modifier les données d'une offre à chiffrer.
La figure 6.2 illustre le graphe de transition des menu de cette tâche.

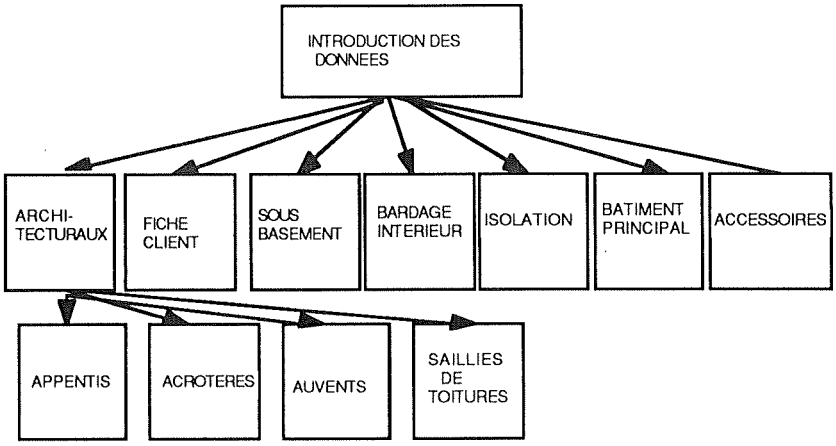


fig 6.2 Graphe de transition des menus

C'est une structure en **arbre**. L'arbre a une profondeur = 3 et une largeur = 7.

Ce sont des valeurs acceptables d'après la recommandation de Miller [Miller,81]. L'utilisateur étant supposé connaître la tâche ne risque pas de se perdre dans le parcours de l'arbre. Même si cela arrivait, une option de retour est disponible qui permet de faire le chemin inverse.

La réorganisation des données a permet de simplifier considérablement l'arbre des menus. En effet, dans la version ancienne, l'arbre avait une profondeur = 3 et une largeur = 17. L'utilisateur a de grandes chances de se perdre. D'autant plus, que les noms des menus ne sont pas toujours claire. Par exemple: "Menu XTRA" où on peut trouver aussi les autostables.

B. Remplissage de formes

Etant donné le nombre élevé de données à introduire, ce style d'interaction se montre très utile. Il permet de regrouper les données dans une seule fenêtre. Ceci évite à l'utilisateur de devoir passer par plusieurs écrans avant de réaliser une sous-tâche. Les données sont présentées dans la fenêtre sous une forme simple claire et spatialement bien délimitée.

Les données similaires sont regroupées entre elles. Les groupes sont séparés par des lignes.

Chaque donnée est représentée par un champs étiqueté par un libellé. Les valeurs sont introduites sous forme alphanumérique. La dimension des champs dépend des données mais chaque champ a une longueur fixe predéterminée.

C.Manipulation directe

La manipulation directe.introduit une amélioration fondamentale par rapport à l'ancienne version du système.

Chaque objet est visualisé et représenté graphiquement. Les couleurs contribuent à différencier les objets entre eux. La figure 6.9 montre l'exemple d'un bâtiment auquel sont accolés un acrotère et un auvent.

L'utilisateur visualise aussi les effets des actions qu'il effectue sur les objets.Il assiste directement à la suppression d'un bâtiment, d'une unité d'isolation... Il constate directement la création d'un nouveau bâtiment.

L'utilisateur a une vue de haut des bâtiments. lorsqu'il travaille sur les murs, ceux-ci sont présentés de face (fig 6.3).

La figure 6.3 montre l'exemple d'un bâtiment auquel sont accolés des architecturaux.

La figure 6.4 illustre le procédes de suppression d'une unité de bardage interieur: la fig 6.4 (a) reprsente l'écran avant la séléction, la fig 6.4 (b), montre la sélection accompagnée d'une demande de confirmation.et la figure 6.4 (c) l'écran après l'exécution de la sélection.

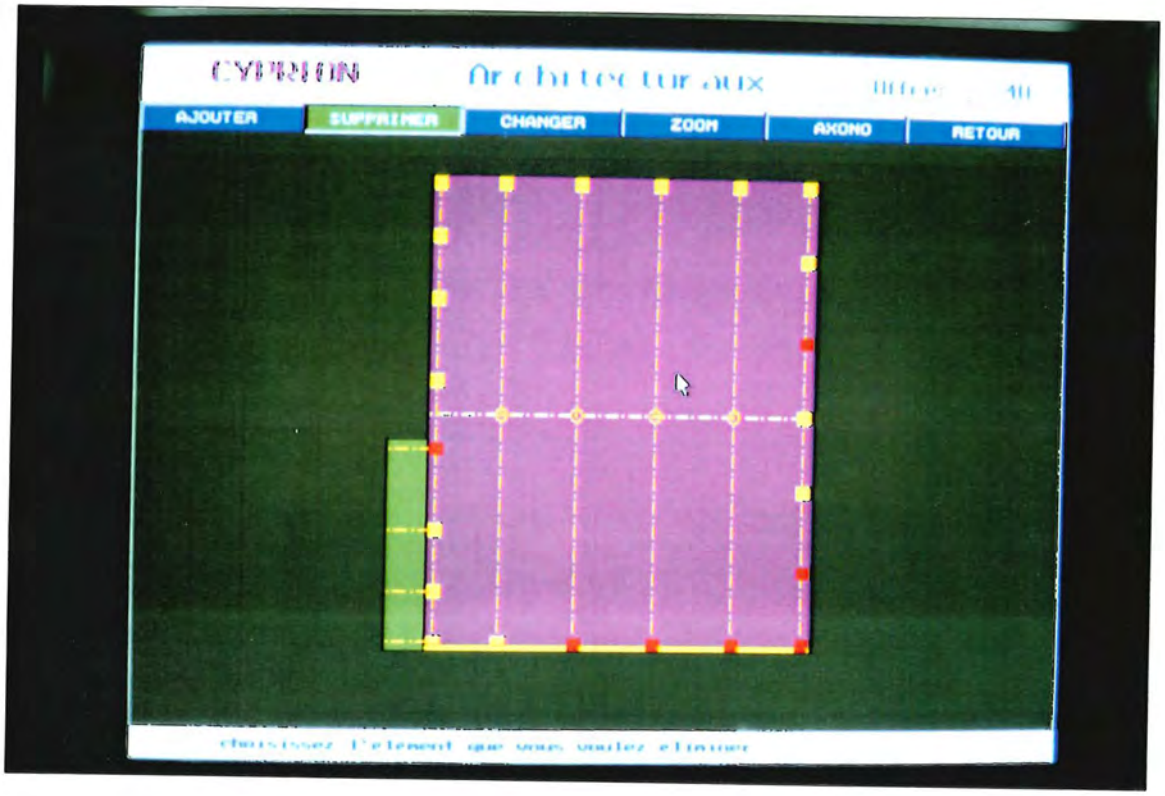


fig 6.3 Bâtiment principal avec archtecturaux

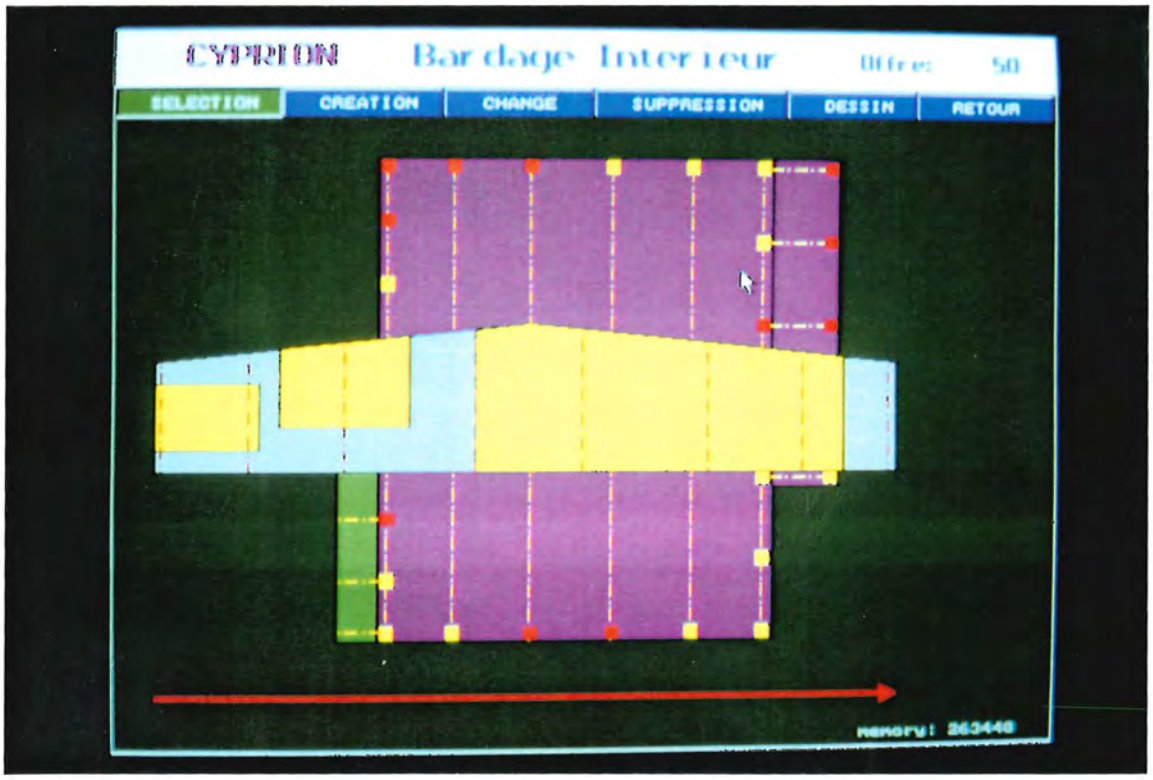


fig 6.4 (a) Dessin des unités de bardage interieur

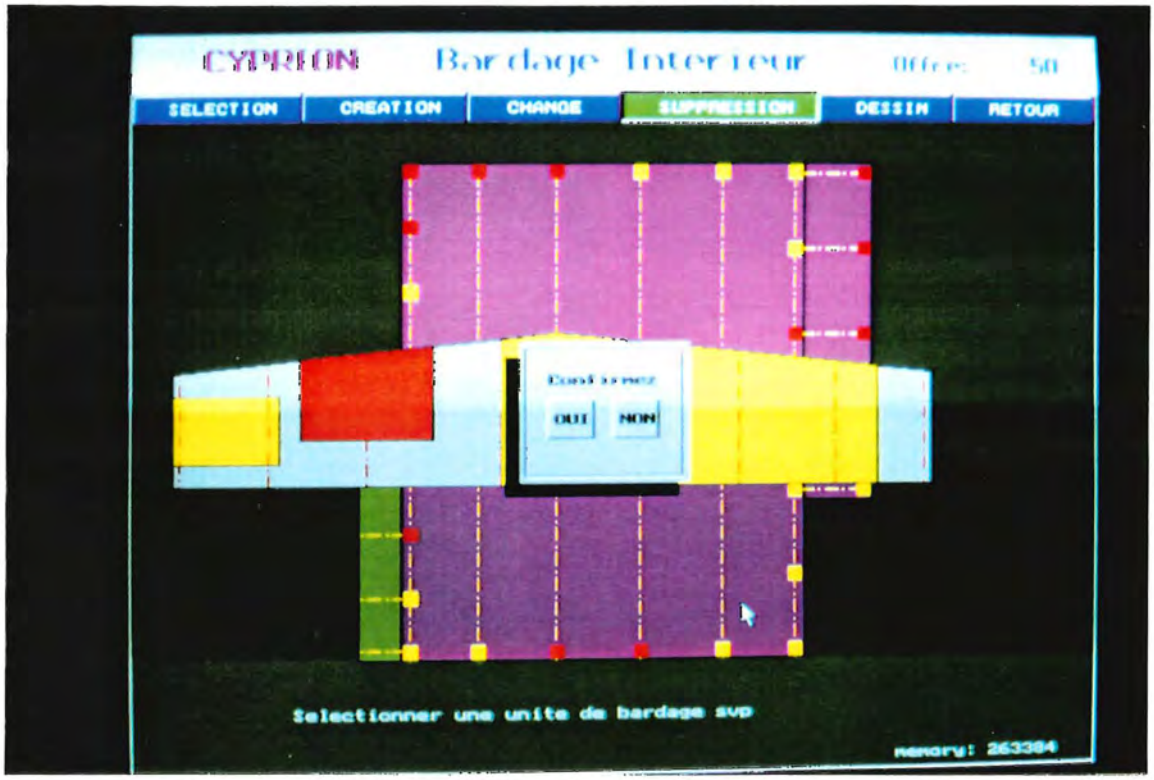


fig 6.4 (b) Sélection d'une unité de bardage intérieur

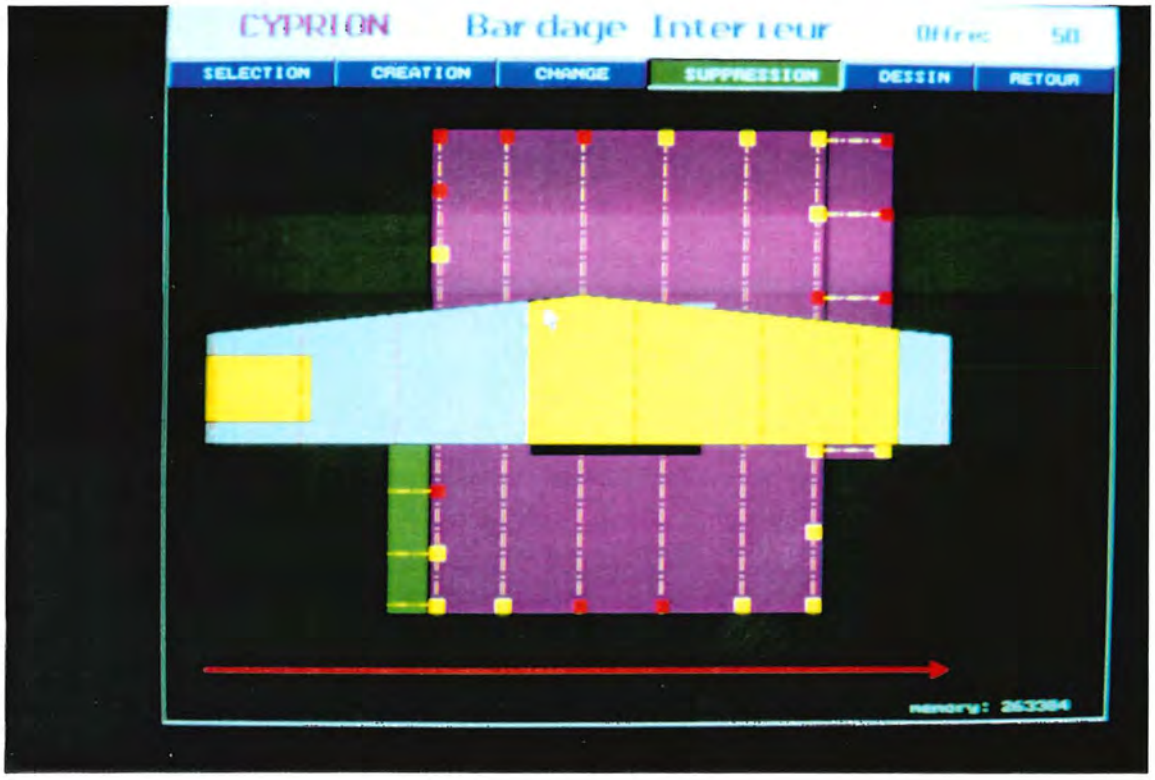


fig 6.4 (c) résultat après la suppression

L'utilisation de la manipulation directe a simplifié le travail de l'utilisateur. Avec la souris, il se positionne et sélectionne l'action et l'objet qui les intéressent. Dans la version ancienne, les bâtiments étaient numérotés. L'utilisateur devait introduire à chaque fois le numéro du bâtiment suivi du numéro du mur surlequel il désire travailler.

Cette facilité de travail se fait sentir encore plus dans le cas de l'introduction des colonnes autostables: dans l'ancienne version, l'utilisateur choisissait d'abord le bâtiment et le mur, puis introduisait les numéro des axes entre lesquels sera positionnée la distribution des colonnes autostables.
Dans la version actuelle, en enfonçant le bouton gauche de la souris, il entoure les colonnes concernées. Le déplacement de la souris engendre une ligne rouge qui visualise ce déplacement. Une fois toutes les colonnes sélectionnées, il lâche le bouton de gauche et enfonce celui de droite. La ligne rouge se referme sur les colonnes sélectionnée. L'utilisateur lâche le bouton la ligne disparait et les colonnes sélectionnée se colorent en rouge. Ce processus est long à décrire mais quasi instantané à l'exécution.

L'utilisation de la manipulation directe a facilité le travail de l'utilisateur, l'a rendu plus simple, plus agréable en minimisant le nombre d'actions nécessaires à la réalisation d'une sous-tâche.

Ceci est très intéressant, mais il ne faut pas oublier que cette facilité d'utilisation cache la difficulté de programmation et le durée de programmation.

VI.2.2.4 Organisation des écrans

Pour raison de cohérence et par soucis de clarté, les écrans sont organisés selon un modèle uniformisé (voir fig 6.5).

Un jeu de couleurs délimite les différentes zones de l'écran.

Un message d'indication s'affiche concernant les données actives dans le but d'orienter l'utilisateur et l'aider dans son choix.

Les données dans la fenêtre sont organisées de telle sorte que la partie supérieur regroupe les données à introduire avec la souris, la partie inférieur celle à introduire du.clavier.

Les unités de mesure sont affichées à gauche du champs de la donnée, ne doivent pas être saisies.

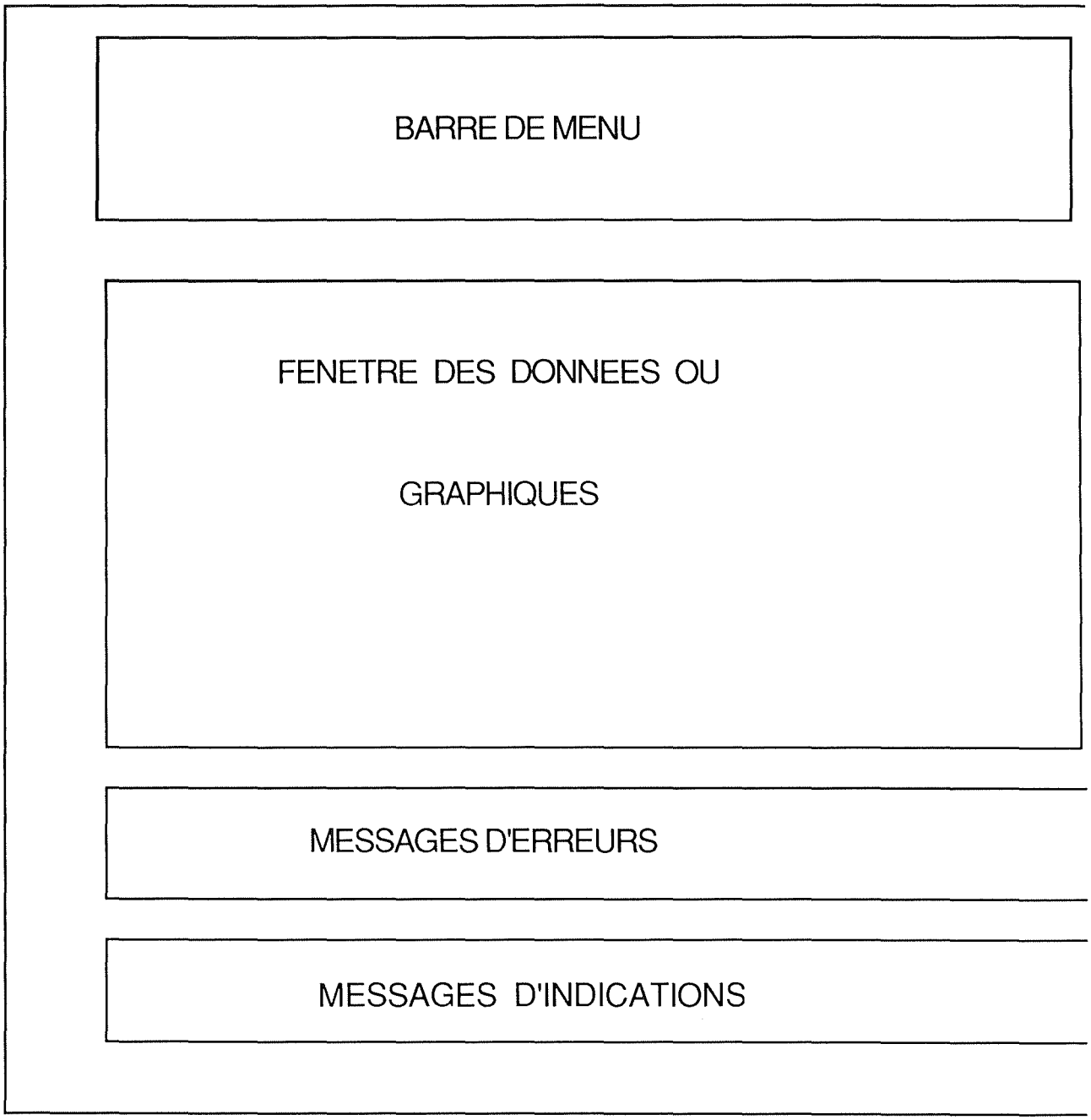


fig 6.5 Organisation des écrans

En résumé, on peut dire que l'utilisateur a devant lui un écran bien organisé, clair et plaisant à voir. La tâche de l'utilisateur est aisée .

VI.2.2.5 Utilisation des couleurs

dans l'interface réalisée, l'utilisation des couleurs occupe une place prépondérante. Elle est à la base de beaucoup de facilités: distinction entre les groupes de données, entre les zones de l'écran, entre les actions actives et non actives ...

Le choix des couleurs doit donc satisfaire d'une part, à ces exigences et, d'autre part, au souci de clarté et d'esthétique. L'utilisation de certaines couleurs, comme le rouge, doit être limitée car c'est une couleur agressive. Elle est souvent synonyme de danger. C'est pourquoi elle a été choisie pour les actions de suppression. Mais parfois, l'utilisation de cette couleur ne peut pas être évitée: quand l'écran regroupe différents objets avec différentes couleurs, le rouge risque d'être la seule couleur convenable qui puisse ressortir parmi les autres. C'est le cas des autostables. Ces colonnes sont dessinées en rouge. Mais, elles sont représentées sous forme de petits carrés. La petitesse de ces carrés limite la vivacité du rouge et le rend moins agressif.Cette décision s'explique par le fait que les combinaisons de couleurs impliquent des couleurs complémentaires et distinctibles à l'oeil en cas de superposition.

Le choix des autres couleurs s'est effectué par essais successifs des couleurs disponibles et aussi par un concensus entre le concepteur et les participants à la conception de l'interface.

VI.3 Analyse ergonomique de l'interface

Cette analyse sera réalisée par rapport aux régles de Shneiderman et aux recommandations de Scapin.que nous avons présentées au chapitre III On verra dans quelle mesure elles sont respectées.

Une remarque importante à faire, est que l'interface réalisée est une **réécriture** de la partie d'introduction des données. Sa conception prend en compte les réactions des utilisateurs vis à vis de l'ancienne version. Un de ses buts sera de satisfaire aux demandes des utilisateurs et de fournir des fonctionnalités qui manquaient comme l'isolation partielle.

VI.3.1 Respect des règles de Shneiderman

R1. Cohérence

La régle R11 est respectée: une sous-tâche commune à une même application ou à plusieurs applications utilise le même plan d'actions pour sa réalisation.

Exemple 1: Sélection des appentis:

Cette sélection s'effectue de manière similaire que ce soit au niveau inter-applications ou intra-application.

*) au niveau intra-application: que ce soit pour une modification ou une suppression, la sélection passe par les mêmes étapes
- l'utilisateur à l'aide de la souris se positionne sur l'appentis de son choix
- il enfonce le bouton gauche de la souris pour valider la sélection

*) au niveau inter-applications: la sélection d'un appentis se fait de manière identique entre l'application correspondant au traitement de l'isolation et celle de traitement des appentis.

Exemple 2:

Considérons les deux sous-tâches suivantes:
- création d'une unité d'isolation
- création d'une unité de bardage intérieur

Dans la suite de l'exemple, une unité d'isolation ou une unité de bardage intérieur sera désignée par UNITE.
Ce sont deux sous-tâches similaires bien que relatives à deux objets différents. La séquence d'actions est la suivante:

1) l'utilisateur choisit de créer une UNITE soit en toiture, soit sur un des murs.

2) l'utilisateur choisit le bâtiment surlequel il veut travailler.

3) Deux possibilités s'offrent alors à l'utilisateur:
- si l'utilisateur a choisi de placer l'unité sur la toiture, la fenêtre d'introduction de données relative au placement d'une UNITE sur la toiture est affichée.
- si l'utilisateur a choisi les murs, il doit sélectionner le mur surlequel il veut travailler. La sélection faite, la fenêtre de création d'une unité sur un mur est affichée.

La régle R12 est aussi respectée que ce soit au niveau de la cohérence lexicale ou de la cohérence spatiale:

a) cohérence lexicale: un objet ou un concept est désigné par le même terme indépendamment du contexte. Ceci permet d'éliminer les incohérences lexicales.

Exemple:

- "RETOUR": terme commun à tout les menus et qui indique comment quitter le menu
- "ESCAPE": la commande qui permet d'interrompre (anormalement) un traitement en cours.

En plus, une nomenclature de termes précis a été choisie. Les dénominations utilisées sont cohérentes avec le vocabulaire de l'utilisateur: ceci n' a pas posé un grand problème car les concepteurs sont des gens de la société et donc connaissent aussi la tâche.
Puisque les concessionnaire de la firme, sont répartis dans plusieurs pays de langues différentes, il a été prévu de traduire les écrans dans les langues de ces pays.Les opérations permises sont désignées par des noms faciles à comprendre du genre: "AJOUTER", "MODIFIER", "SUPPRIMER" ...

b) cohérence spatiale: dans l'analyse fonctionnelle, on a vu que les écrans sont bien organisés: un endroit réservé pour les messages, un autre pour les données... En plus, une donnée commune à plusieurs écrans se retrouve au même endroit indépendamment de l'écran.

Ceci permet un cohérence spatiale entre les différents écrans. L'utilisateur peut de ce fait, anticiper sur les actions physiques. Il peut préparer son oeil à viser à un endroit précis avant même que l'écran change. S'il doit saisir de l'information, par exemple, le temps de saisie est diminué car le temps de recherche pour localiser la donnée est très faible.

R2. La concision:

Elle apparait dans l'utilisation de valeurs par défaut chaque fois que c'est possible. C'est important, car certaines valeurs sont plus affectées aux données que d'autres. La régle R21 est donc respectée.

De même, la règle R23 concernant les abbréviations est respectée. Celles-ci correspondent aux termes courant des utilisateurs.

Par contre, la régle R22 n'est pas En effet, les macros ne sont pas utilisées.

Exemple:

imperméabilité,
type bardage intérieur = PL

R3 Retour d'information et guidage:

La réaction du système aux actions de l'utilisateur est immédiate. Elle est concrétisé par les informations affichées ou les graphiques dessinés. L'écran reflète l'état du système.

Exemple1:

Considérons de nouveau le cas du bardage interieur,une unité nouvellement crée est dessinée dans une couleur différente de la couleur des autres unités.
Que ce soit au niveau de l'introduction du bardage ou de suppression, une légende "--->" apparaît en dessous du dessin (fig 6.4 (a))et indique le sens de calcul et permet de reconnaître le pignon avant du pignon arrière.

Ce retour d'information est informatif. Il réduit la charge informationnelle de l'utilisateur. Celui-ci n'a rien à mémoriser car l'état du système est connu.et montré. Ceci est important car, comme on l'a vu dans le modèle du processeur humain, la mémoire à court terme est limitée.

Le retour d'information et le fait d'informer de manière continue su l'état du système, aident l'utilisateur dans son processus de décision: l'utilisateur effectue une action, l'effet est immédiat et visible, il compare ce nouvel état avec l'état antérieur du système et décide des nouvelles actions à exécuter. Il évalue les résultats (variables physiques) avec ses objectifs (variables psychologiques).

L'utilisateur sait aussi à tout moment où il se trouve dans la séquence d'actions et dans quel menu et quelle application. Ceci grâce d'une part à l'option active qui prend une couleur différents des autre options.

Exemple 2:

Dans la figure.6 4(a), l'item du menu sélectionné, donc "SUPPRESSION" est coloré en vert pour indiquer qu'il est actif. Les autres items sont en bleu.

Exemple 3: Transfert d'un fichier

Le transfert d'un fichier peut s'avérer une opération longue si la taille du fichier est importante. Le temps de transfert ne demeure plus négligeable induisant une longue attente pour l'utilisateur. Cette attente peut être inquiétante pour un utilisateur qui ne connait pas le mécanisme de transfert et le temps qui lui est nécessaire pour s'accomplir. Il peut penser qu'un problème est survenu et décide alors d'interrompre le transfert. Il va relancer le transfert mais pour se retrouver dans la même situation.

Ce problème peut être évité par l'utilisation d'un indicateur de progression. L'utilisateur est constamment rassuré tout au long du transfert. Ce qui se traduit par un sentiment de confort et de satisfaction personnelle (cinquième critère de qualité d'une interface).

Une critique que l'on peut faire à l'interface est illustrée par l'exemple 4.

Exemple 4: Suppression d'un appentis sur lequel sont placés un acrotère et un auvent:

La suppression de cet appentis va engendrer avec elle la suppression de l'acrotère et de l'appentis. Rien n' a été prévu pour informer explicitement de ce phénomène. Une analyse plus fine du problème montre que la tâche n'en est pas moins bien accomplie. En effet, d'un côté, l'utilisateur possède une bonne connaissance de la tâche. D'un autre côté, il a devant lui le graphique des bâtiments où l'acrotère et l'auvent sont explicitement dessinés sur l'appentis. Plus encore, la demande de suppression implique une demande de confirmation. Celle-ci s'accompagne d'une coloration en rouge de tous les bâtiments qui seront affectés par cette suppression: donc l'acrotère et l'auvent aussi. L'utilisateur est donc informé implicitement du problème.

Informer implicitement et non explicitement l'utilisateur ne diminue en rien la qualité ergonomique de l'interface. N'oublions pas que l'utilisateur, donc le concessionnaire ou le client de la société est quelqu'un qui connait la tâche. Introduire une demande de confirmation pour l'auvent et l'acrotère peut résulter en un sentiment de frustration, d'impatience et d'énervement de la part de l'utilisateur. Le temps d'exécution sera aussi prolongé.
On peut dire que le fait d'informer implicitement l'utilisateur est une qualité ergonomique dans le cas de cet interface: la connaissance de l'utilisateur est prise en compte et respectée.

R4 Gestion des erreurs

Toute valeur introduite est immédiatement validée. S'il y a erreur, celle-ci est détectée et un message d'erreur est affiché à l'écran. Le signalement de l'erreur se manifeste par un signal sonore accompagné d'un message explicatif en bas de l'écran. Le message d'erreur renseigne l'utilisateur sur la nature de l'erreur et la correction a apporter.

La régle R41 sur le signalement d'erreur est donc respéctée.

En ce qui concerne la réversibilité des actions rien n' a été prévu. Il n'y a pas de fonctions disponibles qui permettent d'annuler l'effet d'une action terminée et de retourner à l'état antérieur du système. Il n'y a pas de gestion de l'historique des variables et des actions. Ceci peut être gênant pour l'utilisateur comme l'illustre l'exemple suivant:

Exemple 2:

Après une succession de modifications, l'utilisateur définit l'utilisateur d'isolation qui satisfait à ses exigences. Pendant un traitement ultérieur,

il supprime par mégarde cette unité. Il essaie de se souvenir des actions successives pour la redéfinir. En vain, sa mémoire à court terme limitée en durée, peut lui faire défaut.

Ce genre de mésaventures peut être grave et très gênant pour l'utilisateur. Sans oublier la perte de temps que cela entraîne. Même s'il se souvient avec exactitude des actions nécessaires, la réexécution va prendre un certain temps sans oublier l'effort cognitif qu'il doit fournir pour se souvenir de ces actions.

Pour remédier à ce problème, une demande de confirmation dans le cas de suppression a été prévue. Mais ce n'est pas suffisant. D'autant plus que la même situation peut arriver avec la modification des valeurs.

Une suggestion serait de définir une fonction "défaire" ("undo" en anglais") qui annulle l'effet de l'action exécutée en dernier lieu. Permettant ainsi de revenir à l'état précédent l'exécution de l'action.

Ce genre de fonction permet à l'utilisateur d'être plus confiant dans son travail mais aussi d'accélérer la réalisation des tâches. En effet; si les variables gardent un historique de leurs états antérieurs, il est plus simple et plus rapide de retourner à ces états que de recalculer leurs valeurs et réexécuter les actions.

R5. La flexibilité

Ce critère n'a pas été pris en considération. L'interface n'est pas ajustable à des environnements variés et à des populations d'utilisateurs diverses. L'utilisateur est guidé et n'a pas l'initiative du choix. Ceci peut être amélioré par l'utilisation de raccourcis et de macrocommandes. L'utilisateur pourra alors passer au menu qu'il souhaite directement ou bien exécuter une action sans devoir passer par le chemin habituel qui risque d'être long. Ceci va accélérer considérablement le processus d'exécution.

Exemple: Copie d'une offre

Cette fonction permet de dupliquer une offre. La nouvelle offre aura un numéro d'offre identifiant différent de l'autre, mais aura toutes les autres caractéristiques.identiques. Cette opération est intéressante lorsque deux offres diffèrent seulement par des petits détails, par exemple la répartition des autostables. Au lieu que l'utilisateur recrée objet par objet, il duplique l'offre, puis apporte les modifications appropriées. Ceci diminue considérablement le temps de traitement et facilite le travail de l'utilisateur. Cette fonctionnalité doit s'avérer pratiquement très agréable dans la mesure où le vendeur peut faire des offres "proches" sans trop de manipulation.Pour un expert, cette opération peut être encore plus améliorée par l'utilisation de macrocommandes. L'utilisateur peut définir dans des macrocommandes des chemins les plus empruntés pour la modification des offres

dupliquées. La macrocommande déclenchera alors les actions successives et l'utilisateur pourra effectuer les modifications qu'il souhaite sans presque aucun effort. Si plusieurs opérations sont prévues, il peut aussi utiliser des raccourcis pour parcourir plus rapidement les menus.

La seule forme de flexibilité que l'on peut relever est l'adaptation de la terminologie à la langue utilisée par l'utilisateur: c'est la flexibilité linguistiques.

En résumé, l'on peut dire que l'interface satisfait et respecte les régles concernant la cohérence, le retour de l'information et la gestion des erreurs. La flexibilité est partiellement respectée.
Certaines améliorations peuvent être apportées: elles concernent surtout l'utilisation de fonctions telles que le "défaire" et les macrocommandes.

VI.4.2 Respects des recommandations de Scapin

On ne traitera ici que les règles qui ne sont pas couvertes par les règles empiriques.

E1. Mode d'entrée

A chaque libellé de donnée est associé un champ d'édition dont la valeur est une suite du caractère "blanc". L'introduction des valeurs se fait par remplacement successifs de ces blancs.

E2. changement de mode

Le passage du mode souris au mode clavier est très limité du fait de l'organisation de l'écran (voir analyse fonctionnelle).

E3. Actions minimales

Toutes les données dérivables sont calculées automatiquement par le logiciel et ne sont pas demandées à l'utilisateur.

Exemple

- calcul de la hauteur libre
- calcul des distributions des colonnes de pignons.

E12. Interruption du dialogue

A tout moment , l'utilisateur peut décider d'interrompre le dialogue. avec la fonction "ESCAPE".

Les autres recommandations sont respectées. Elles ont été traité dans les analyses précédentes.

Résumé

l'interface est structurée en sous-tâches qui correspondent aux objectifs de l'utilisateur et à la découpe réelle de la tâche.
Les sous-tâches sont gérées de manière cohérente, que ce soit au niveau d'une application ou entre les applications.
Une terminologie adéquate et consistante est utilisée pour dénommer les différents concepts de l'interface. Elle correspond aux dénominations de l'utilisateur.

L'utilisation de valeurs par défaut permet de minimiser l'effort cognitif que doit fournir l'utilisateur pour mémoriser les informations. Elles prolongent la mémoire à court terme de l'opérateur humain.
L'utilisateur est assisté tout au long de son travail. Chaque valeur de donnée introduite est immédiatement validée. S'il y a erreur, l'utilisateur en est avertis par un signal sonore. Ce signal s'accompagne d'un message d'erreur qui s'affiche dans une zone fixe de l'écran. Le message explicite le type d'erreur et le genre de correction à y apporter.
L'utilisateur se voit constamment rassuré sur l'évolution de son travail. Grâce à un jeu de couleurs bien choisi, il sait à tout moment quelle action il a activé ou quelle donnée il a modifié. L'effet de chaque action est immédiatement retourné soit sous forme de données affichées, soit sous forme de dessins. Même s'il quitte son poste de travail pour un temps long, il peut sans problème retrouver le fil de son travail et reprendre la séquence d'actions là où il l'a laissée. Ce retour d'information guide et oriente l'utilisateur dans la réalisation des objectifs qu'il s'est fixés.

Le caractère ergonomique de l'interface est indéniable. Les principaux critères et recommandations sont respectés. L'interface tient compte des limites des capacités cognitives humaines. Son but premier est de satisfaire l'utilisateur.

Les améliorations qui peuvent encore être faites concernent surtout l'utilisation de raccourcis pour rendre l'interface adaptable à des utilisateurs experts. La réversibilité des actions aussi pourrait être ajoutée aux autres fonctionnalités de l'interface.

Jusque là, on a analysé le caractère ergonomique de l'interface. Mais une interface qui satisfait aux critères d'ergonomie n'est pas nécessairement conviviale et attirante pour l'utilisateur.
La convivialité étant un caractère très recherché par l'utilisateur, la section suivante étudie de plus près cette notion.

V1.4 Mesure de la convivialité de l'interface

Ce sont des mesures qui devraient être évaluées *a posteriori* par les utilisateurs de l'interface. Mais, étant donné le fait que l'interface n'est pas encore opérationnelle puisqu'il reste des fonctionnalités à terminer (ponts roulants, mezzanines...), les mesures qui seront données ci-après proviennent de ma propre expérience renforcée par les réactions de personnes qui ont pu observer l'interface lorsqu'elle était en cours de développement. Ce sont soit des personnes du département informatique, soit appartenant à d'autres départements de la société et qui sont directement ou indirectement concernés par l'interface. Par exemple, les responsables du département de Marketing. Il y a aussi les employés de la société qui sont délégués auprès des concessionnaires dans les différents pays.

Ces mesures sont relatives aux critères de qualité d'une interface de Shneiderman [Shneiderman,87]:

1. Temps d'apprentissage

L'interface utilisant la sélection des menus, le remplissage de formes et la manipulation directe, l'apprentissage des fonctionnalités de l'interface est très rapide et ne pose pas de problème particulier. D'autant plus que, les utilisateurs, en général connaissent la tâche et sont familiarisés avec l'ancien système.

2. Rapidité d'exécution

Le temps d'exécution d'une tâche dépend du type de l'ordinateur utilisé. J'ai réalisé mon travail sur un micro ordinateur de type COMPAQ 386 dont la rapidité d'exécution est étonnante: le temps de réponse est dans ce cas est inférieur à 2 secondes. Par compte, cette rapidité est plus modérée sur d'autres type de machines. Mais de toute façon, l'exécution est rapide et ne dépasse pas les limites d'acceptabilité.

3. Taux d'erreurs

Les erreurs sont très limitées, car l'utilisateur est constamment guidé dans son travail. Les erreurs qui peuvent survenir sont surtout des erreurs de frappes ou causés par un relâchement hâtif du bouton de la souris. Il faut signaler que la vitesse de la souris est adapté à celle de déplacement de la main de l'utilisateur.

4. Période de rémanence

Le séquencement des actions associées aux sous-tâche est cohérent avec les actions réelles et les objectifs de l'utilisateur, le danger de perte d'information acquise est quasi nul.

5. Satisfaction personnelle

Il a été constaté que, tous ceux qui ont vu le système ont été très satisfaits par les facilités et les fonctionnalités qu'il leur offre. La présentation des écrans, l'utilisation des couleurs et des graphiques étaient hautement appréciées.

Selon notre point de vue, l'interface réalisée se révèle ergonomique, conviviale et très agréable à l'utilisation. Elle rencontre tous les objectifs fixés.

CONCLUSION

La synthèse exposée dans ce mémoire met en évidence la nécessité de mettre en commun les compétences de l'ergonome et de l'informaticien.

L'apport des sciences cognitives apparaît sous deux approches: une approche théorique et une approche issue de l'expérimentation.

Dans l'approche théorique, on remarque deux théorie fondamentales: un théorie de l'action et le modèle du processeur humain.

La théorie de l'action est un modèle explicatif du comportement humain. Il distingue le monde psychologique de l'utilisateur et le monde physique de la tâche, il montre la différence entre l'état réel du système et l'état perçu de l'utilisateur. Ce modèle aide le concepteur à identifier les besoins de l'utilisateur et à comprendre la nature de sa tâche.

Le modèle du processeur humain voit l'être humain comme un processeur d'information. Il montre les limites des capacités cognitives humaines et la nécessité d'y suppléer.

L'approche théorique essaie d'expliquer les mécanismes qui régissent le comportement de l'être humain.

L'approche expérimentale est par contre un ensemble de principes et recommandations ergonomiques issus de l'observation et de l'expérience. Il constitue un guide ergonomique pour le concepteur.

Chacune de ces théories a ses lacunes. La théorie de l'action explique le comportement humain mais ne s'appuie sur aucun cadre formel. Par contre, le modèle du processeur humain, formalise les performances humaines dans un vocabulaire proche de celui de l'informaticien mais, il n'explique pas les processus cognitifs les plus fondamentaux tel l'apprentissage.

Les principes pratiques traitent de cas bien particuliers et sont souvent issus d'applications spécifiques.

En fait, la principale limitation des ces théories proviennent de ce qu'elles ne fournissent pas une méthode de conception sur laquelle le concepteur peut fonder son travail. Mais, elles montrent un point fondamental qui est la nécessité pour le concepteur de bien comprendre l'utilisateur et la nature de sa tâche.

Ces faits ce sont vus confirmés lors de la réalisation de l'interface d'une application interactive pour la construction de bâtiments professionnels nommée Cyprion, dans le cadre d'un stage.

En effet, le premier problème auquel nous avons été confronté est justement la compréhension de la nature de la tâche. La première phase de mon projet était donc une analyse de la tâche: comprendre les concepts et objets véhiculés et élaborer un schéma entité/association qui définit l'ensemble des objets et les associations auxquelles ils contribuent. La difficulté de cette phase provenait

d'une part du fait que, la nature de la tâche nous paraissait totalement étrangère et, d'autre part, du manque de documentation.
La deuxième phase avait pour but l'identification des fonctionnalités requises, ce qui correspond à l'identification des besoins de l'utilisateur.
Pour réaliser ce but, nous sommes partis du logiciel existant et des explications fournies par les responsables du projet Cyprion. Nous avons du faire face à un manque crucial de documentation.
La phase suivante concerne la restructuration des données pour pouvoir satisfaire les fonctionnalités espérées. Cette phase a mené à la détermination des structures de données nécessaires à la programmation.
La première première phase interactive à savoir l'introduction des données du bâtiment principal, a été le premier objet de notre implémentation. Tout d'abord, il a fallu concevoir la fenêtre des données; regrouper les données, positionner les champs et les libellés... L'écran final se devrait d'être clair et présentable. La fenêtre finale était aboutie après de nombreux essais de localisation des données. L'élaboration de la fenêtre peut paraître simple et facile, mais en réalité, c'est ajustements demandent du temps et nécessitent beaucoup de programmation. Mais la réalisation de cette fenêtre nous a permis d'acquérir une stratégie à suivre pour concevoir les autres.

Nous avons pu tiré certaines conclusions:

- concevoir une interface évoluée nécessite du temps et une certaine habileté dans la programmation qui ne peut être acquise qu'avec l'expérience.
- les choix initiaux n'étaient appuyés par aucun principe ou règle ergonomique. En fait, le présent travail a été réalisé *a posteriori* après la réalisation de l'interface.

L'évaluation ergonomique de l'interface a pourtant montré que celle-ci respecte les critères ergonomiques et satisfait aux exigences de qualité de convivialité.

Les choix étaient opérés de manière intuitive mais aussi conditionnés par nos connaissances préliminaires, notamment celles du Macintosh. Mais, plus important encore, est le fait que notre travail.n'était pas isolé: il s'inscrivait au sein d'une équipe responsable de tout le projet Cyprion. Notre travail était donc conditionné par le leur et s'en inspirait.

Les trois points évoqués nous permettent de conclure que le travail d'un concepteur est influencé par ses connaissances, son expérience et par l'environnement de travail. La connaissance des principes cognitifs sensibilise l'informaticien sur certains points critiques et oriente sa réflexion et son travail, mais ne l'aide pas dans la conception de l'interface. Elle lui permet pourtant de vérifier le caractère ergonomique et d'entreprendre des modifications s'il le faut. Elle permet aussi de faciliter son travail en l'aidant à structurer son travail en fonction de ces critères et lui éviter ainsi une recherche exhaustive d'une solution

acceptable. En effet, le fait de connaître ces principes ergonomiques *a priori* avant d'entamer notre travail, nous aurait permis d'opter plus rapidement pour certaines solutions telles la disposition des données,.l'utilisation de couleurs...Cela se serait traduit par un gain de temps appréciable. Pourtant, nous doutons que l'interface ainsi réalisée eût été radicalement différente.

- une autre remarque concerne l'environnement organisationnel. En effet le travail de l'informaticien est soumis à des contraintes économiques, techniques et sociales.

La prise en compte de concepts cognitifs, de manière implicite ou explicite, est un facteur important dans la conception d'interfaces conviviales et ergonomiques. Mais l'apport est limité car ils ne fournissent pas une méthode de conception. Le concepteur doit puiser dans ses connaissances, son expérience, son imagination et sa créativité pour réaliser les résultats souhaités.

[Bodart,89] BODART, F., PIGNEUR Y., *conception assistée des systèmes d'information: méthodes-modèles-outils*, 2e édition MASSON 1989.

[Bodart] BODART, F., VANDERDONCKT, J., *La prise en compte de facteurs ergonomiques dans la conception des interfaces homme-machine*, F.U.N.D.P

[Card,83] CARD, S.K, MORAN,T.P et NEWELL,A., *the psychology of human computer interaction*, Lawrence Elbaum Associates, Lawrence Erlbaum Assocites, Hillsdale (New Jersey), 1983.

[Coutaz,88] COUTAZ,J., *interface homme-ordinateur: conception et réalisation*

[Dumas,88] JOSEPH S. DUMAS, *designing user interface for software* Prentice Hall 88

[Hollan,89] HOLLAN, J.D., HUTCHINS, E.L.et NORMAN, D.A., *Direct Manipulation Interfaces,* in *User Centred sysytems Design*, Norman, D.A. et Draper, S.W. (éds), Lawrence Erlbaum Associates, New Jersey, 1986, p 87-124.

[Miller,81] MILLER, DWIGHT,P., *The debth/breadth trade-off in hierarchical computer menus,* Proc. of the 25th Annual Meeting of the Human Factors Society, 1981, pp.286-300

[Norman 86] DONALD A. NORMAN et S.W DRAPPER, *User Centred System Design: New Perspectives on Human-Computer Interaction*, Lawrence Elbaum, Hills Dale (New Jersey), 1986

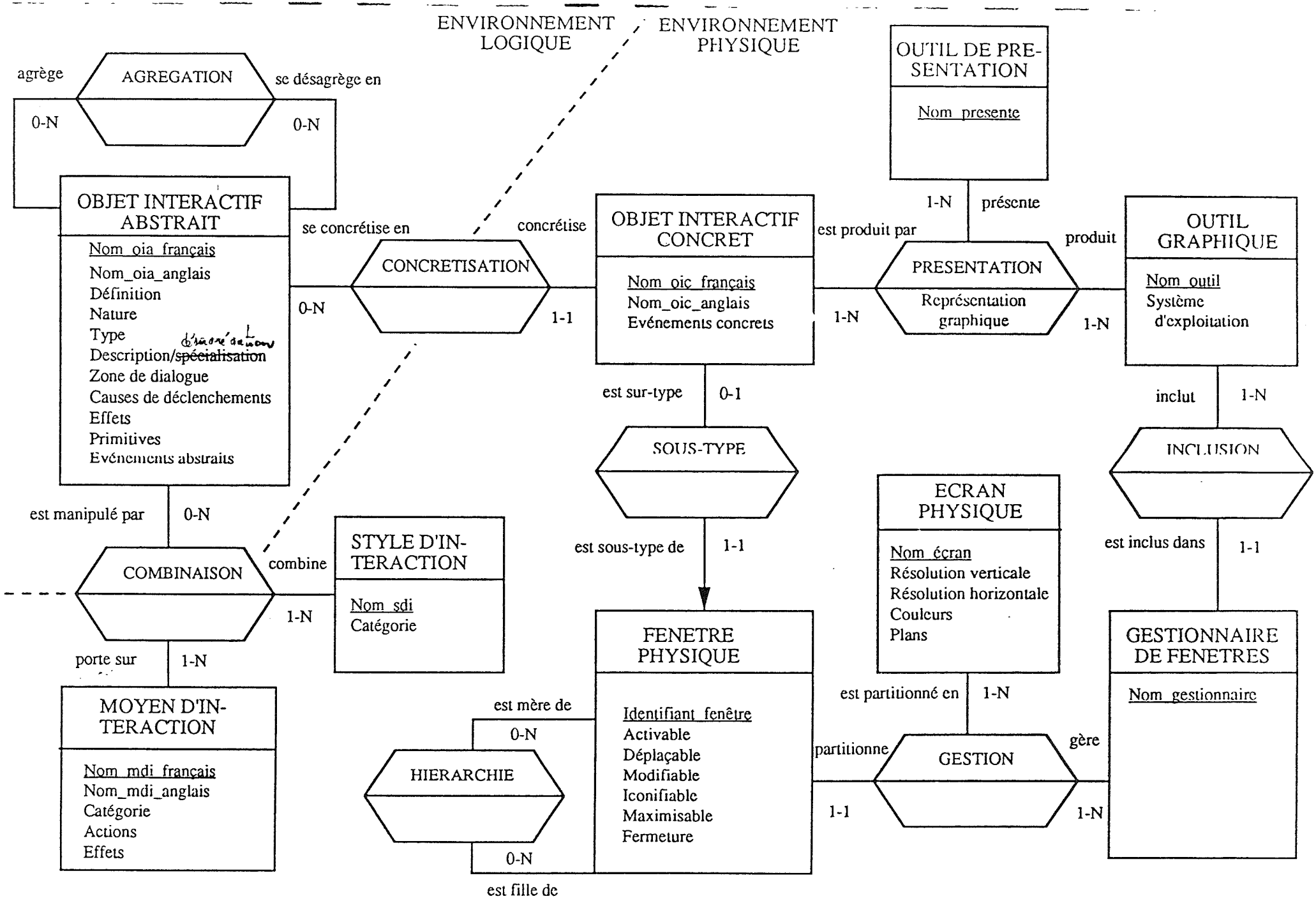
[Provot,90] PROVOT, I. et VANDERDONCKT, J, *Les objets interactifs: classification et typologie*

[Scapin,87] SCAPIN, D.L, *Guide ergonomique de conception des interfaces homme-ordinateur,* INRIA (Institut National de Recherche en Informatique et en Automatique), Rocquencourt (France), 1984.

[Shneiderman 87] SHNEIDERMAN, B., *Designing the user interface: strategies for effective human-computer interaction*, Addison Wesley, Reading (Massachussetts), 1987.

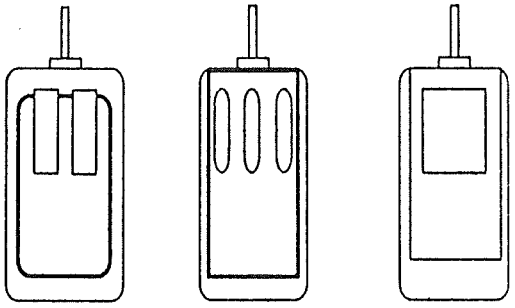
[Young,82] YOUNG, R.M, and HULL, A., Cognitive aspects in the selection of VIWEDATA options by casual users, pathways to the Information Society, Proc. 6th International Conference on Computer Communication (London), Septembre 1982.

ANNEXE A


ENVIRONNEMENT LOGIQUE
ENVIRONNEMENT PHYSIQUE
agrège
AGREGATION
se désagrège en
0-N
0-N
OBJET INTERACTIF ABSTRAIT
Nom_oia_français
Nom_oia_anglais
Définition
Nature
Type
Description/spécialisation
Zone de dialogue
Causes de déclenchements
Effets
Primitives
Evénements abstraits
se concrétise en
CONCRETISATION
concrétise
0-N
1-1
OBJET INTERACTIF CONCRET
Nom_oic_français
Nom_oic_anglais
Evénements concrets
est produit par
PRESENTATION
Représentation graphique
produit
1-N
1-N
OUTIL DE PRE-SENTATION
Nom_presente
1-N
présente
OUTIL GRAPHIQUE
Nom_outil
Système d'exploitation
est sur-type
0-1
SOUS-TYPE
est sous-type de
1-1
inclut
1-N
INCLUSION
est inclus dans
1-1
est manipulé par
0-N
COMBINAISON
combine
1-N
STYLE D'IN-TERACTION
Nom_sdi
Catégorie
porte sur
1-N
MOYEN D'IN-TERACTION
Nom_mdi_français
Nom_mdi_anglais
Catégorie
Actions
Effets
ECRAN PHYSIQUE
Nom_écran
Résolution verticale
Résolution horizontale
Couleurs
Plans
est partitionné en
1-N
partitionne
1-1
GESTION
gère
1-N
GESTIONNAIRE DE FENETRES
Nom_gestionnaire
FENETRE PHYSIQUE
Identifiant_fenêtre
Activable
Déplaçable
Modifiable
Iconifiable
Maximisable
Fermeture
est mère de
0-N
HIERARCHIE
0-N
est fille de

3. DESCRIPTION DES MOYENS D'INTERACTION

Nom français : **Souris**
Nom anglais : Mouse



Définition : dispositif servant à déplacer le curseur ou le pointeur à travers l'écran pour effectuer des sélections ou réaliser des opérations graphiques telles que déplacer une fenêtre.

Catégorie: moyen d'interaction indirect.
Actions:

1. déplacement du pointeur : bouger la souris sans appuyer sur un bouton, ce qui se traduit par une nouvelle localisation du pointeur et une modification éventuelle de l'aspect du pointeur (la forme du pointeur peut varier en fonction de sa position sur l'écran);

2. pression : appuyer sur un des boutons de la souris et maintenir la pression un certain temps tandis que la souris reste stationnaire

3. clic : presser un bouton et le relâcher immédiatement sans bouger la souris

4. double clic : cliquer rapidement deux fois de suite

5. glissement ("Dragging") : presser un bouton tout en déplaçant la souris

Nom français : **Suiveur**
Nom anglais : Trackball

Définition: dispositif de localisation servant à déplacer ou à pointer un curseur sur l'écran par rotation d'une boule dans la direction adéquate. Le dispositif reste fixe sur la surface de travail : seule la boule est mobile.

Catégorie: moyen d'interaction indirect.
Actions :

Nom français: **Manette**
Nom anglais : Joystick

Définition: dispositif de localisation servant à déplacer ou à pointer un curseur sur l'écran par orientation d'un manche de façon horizontale (vers la gauche ou vers la droite) et/ou verticale (vers le haut ou vers le bas).

Catégorie: moyen d'interaction indirect.
Actions : identiques à celle de la souris

Nom français : **Réticule**
Nom anglais :

Définition: dispositif combinant deux molettes rotationnelles, chaque molette indiquant le déplacement ou la position d'un curseur selon un axe de l'écran.

Catégorie: moyen d'interaction indirect.
Actions :

1. déplacement horizontal du pointeur : faire rouler la molette horizontale, ce qui déplace le pointeur horizontalement et peut engendrer une modification éventuelle de l'aspect du pointeur;

2. déplacement vertical du pointeur : faire rouler la molette verticale, ce qui déplace le pointeur verticalement et peut engendrer une modification éventuelle de l'aspect du pointeur;

Nom français : **Tablette graphique**
Nom anglais : Graphics table

Définition : stylet ou curseur manuel (éventuellement garni de boutons) dont le déplacement ou le positionnement sur une surface plane traduit un déplacement ou un positionnement du curseur à l'écran.

Catégorie: moyen d'interaction indirect.
Actions :

1. déplacement du pointeur : déplacer le stylet sans appuyer sur aucun bouton, ce qui se traduit par une nouvelle localisation du pointeur et une modification éventuelle de l'aspect du pointeur;

2. pression : appuyer sur un des boutons du stylet ou de la tablette en le maintenant enfoncé

3. clic : presser un bouton et le relâcher immédiatement sans bouger le stylet

4. double clic : appuyer deux fois de suite rapidement sur un bouton du stylet

5. glissement : presser un bouton tout en déplaçant la souris

Nom français : **Tablette à résonance**
Nom anglais : Sonic table

Définition : stylet dont le déplacement ou le positionnement sur une surface rectangulaire produit deux ondes sonores reçues par deux microphones latéraux pour traduire le déplacement ou le posit ionnement du curseur à l'écran.

Catégorie: moyen d'interaction indirect.
Actions : identiques à la tablette graphique

Nom français : **Crayon optique**
Nom anglais : Optical reader

Définition : dispositif en forme de crayon permettant à l'utilisateur de désigner un point, une ligne, une surface sur l'écran soit par fibre optique, soit par extrémité sensitive.

Catégorie: moyen d'interaction direct.

Actions :

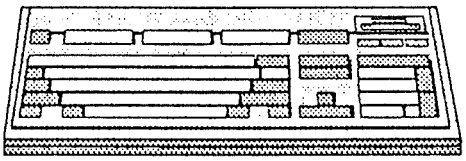
1. pression : appuyer sur le panneau mobile du crayon et maintenir la pression un certain temps;
2. pointage : presser le panneau du crayon et le relâcher immédiatement;
3. glissement : presser le panneau du crayon tout en déplaçant ce dernier.

Nom français : **Ecran tactile**
Nom anglais : Touch screen

Définition: dispositif permettant à l'opérateur de désigner un objet sur l'écran par simple positionnement de son doigt.

Catégorie: moyen d'interaction direct.
Actions :

Nom français: **Clavier**
Nom anglais : Keyboard



Définition : dispositif (électro)mécanique destiné à la saisie d'informations, au pointage et au déplacement d'un curseur via pression de touches.

Catégorie: moyen d'interaction direct.
Actions : pression d'une touche ou d'une combinaison simultanée de touches

ANNEXE B

```
        /* Programme Principal */

#include <conio.h>
#include <stdio.h>
#include <alloc.h>
#include <process.h>
#include <graphics.h>
#include <dos.h>
#include <stdlib.h>

#include "\cyprion\mouse.h"

#define extern unsigned _stklen = 10000

void Initialize(void);

typedef struct
{
 char *text[16];
 char *act[15];
 int  mx[16],my[16];
 int  item,nb;
 int  color,colora;
} POP_MENU;

 void     *image = (char *)0;
 POP_MENU  *pm;
int main()
{
 int  ip,mode,dy;
 char action[20];
 int  item_nb = 0,n;
 Initialize();
 mycolors();
 cleardevice();
 cyprion();
 draw_title();
 dy = textheight("P") + 10;
 get_menus(dy);
 n = 0;
 pm[0].item = -1;
 draw_menu(&pm[0],dy);
 hello_mickey();
 while(1)
  {
   drive_menu(&pm[n],&item_nb,dy);
```

```
if (item_nb != -1)
 {
  strcpy(action,pm[n].act[item_nb]);
  if (strncmp(action,"$M",2) == 0)
     {
      sscanf(&action[2],"%d",&n);
      pm[n].item = -1;
      get_image(&pm[n],dy);
      draw_menu(&pm[n],dy);
     }
  else if (strcmp(action,"EXIT") == 0)
     {
      closegraph();
      restorecrtmode();
      clrscr();
      exit(0);
     }
  else
     {
      char *args[] = {"calling program","from-menu","X",NULL};
      mode = getgraphmode();
      if (strncmp(action,"EMU",3) == 0)
        {
         system("cd\\EMU-TEK");
        restorecrtmode();
         system("EMU-TEK CH");
         system("CD\\TC");
         set_mouse_position(320,240);
        }
      else if (strcmp(action,"C:\\TC\\ISOL") == 0)
            {
              strcpy(action,"C:\\TC\\ISOBARD");
              args[2] = "1";
            ip = spawnv(P_WAIT,action,args);
            }
     else if (strcmp(action,"C:\\TC\\BARD") == 0)
       {
         strcpy(action,"C:\\TC\\ISOBARD");
         args[2] = "2";
        ip = spawnv(P_WAIT,action,args);
       }
      else if (strcmp(action,"C:\\TC\\SOUSBAS") == 0)
       {
         strcpy(action,"C:\\TC\\ISOBARD");
         args[2] = "3";
        ip = spawnv(P_WAIT,action,args);
```

```
          }
         else
         ip = spawnv(P_WAIT,action,args);
         setgraphmode(mode);
         mycolors();
         draw_title();
         draw_menu(&pm[0],dy);
         if (n) draw_menu(&pm[n],dy);
        }
     }
   else if (n != 0)
     {
      reset_image(&pm[n],n);
      n = 0;
/*       pm[n].item = -1; */
     }
  }
}
/*=================================================================
=====*/
draw_menu(pm,dy)
 POP_MENU *pm;
 int      dy;
 {
  int  i,ni,color;
  ni = pm->nb;
  for (i = 0;i < ni;i++)
    {
     color = pm->item == i ? pm->colora : pm->color;
     draw_box_menu(pm,i,dy,color);
    }
 }
/*************************************************************************/
draw_box_menu(pm,nb,dy,color)
int    ` nb,dy,color;
POP_MENU *pm;
{
 setcolor(WHITE);
 setfillstyle(1,color);
 bar3d(pm->mx[nb],pm->my[nb],pm->mx[nb]+textwidth(pm->text[nb]),
      pm->my[nb]+dy,0,0);
 setcolor(BLACK);
 if (color == pm->colora)
   {
    moveto(pm->mx[nb]+textwidth(pm->text[nb]),pm->my[nb]);
    lineto(pm->mx[nb],pm->my[nb]);
```

```
   lineto(pm->mx[nb],pm->my[nb]+dy);
  }
 else
  {
   moveto(pm->mx[nb]+textwidth(pm->text[nb]),pm->my[nb]);
   lineto(pm->mx[nb]+textwidth(pm->text[nb]),pm->my[nb]+dy);
   lineto(pm->mx[nb],pm->my[nb]+dy);
  }
 setcolor(WHITE);
 outtextxy(pm->mx[nb]+1,pm->my[nb]+6,pm->text[nb]);
}
/*****************************************************************************/
drive_menu(pm,item,dy)
POP_MENU  *pm;
int       *item,dy;
 {
  int i,iold,in_box;
  int bmouse,xmouse,ymouse;

  iold = pm->item;

  mouse_display();
  bmouse = 0;
  while(bmouse == 0)
   get_mouse(&bmouse,&xmouse,&ymouse);
  while(1)
    {
     mouse_display();
     get_mouse(&bmouse,&xmouse,&ymouse);
     if (bmouse == 1)
        {
         in_box = 0;
         for (i = 0;i < pm->nb;i++)
          {
            if (xmouse >= pm->mx[i] && xmouse <= pm->mx[i]+textwidth(pm-
>text[i]) &&
                ymouse >= pm->my[i] && ymouse <= pm->my[i]+dy)
             {
              in_box = 1;
              if (i != iold)
                 {
                  mouse_hide();
                  if (iold != -1)
                    {
                     draw_box_menu(pm,iold,dy,pm->color);
                    }
```

```
                draw_box_menu(pm,i,dy,pm->colora);
                iold = i;
               }
             }
           }
         if (!in_box  && iold != -1)
          {
           mouse_hide();
           draw_box_menu(pm,iold,dy,pm->color);
           iold = -1;
           pm->item = -1;
          }
        }
     else if (bmouse == 2)
        {
         mouse_hide();
         *item = -1;
         while (bmouse != 0)
           {get_mouse(&bmouse,&xmouse,&ymouse);}
         return;
        }
    else if (bmouse == 0 && iold != -1)
      {
        mouse_hide();
        *item = iold;
        pm->item = iold;
        return;
      }
   }
}
/*********************************************************************/
get_image(pm,dy)
POP_MENU *pm;
int     dy;
{
 int x0,y0,x1,y1;
 unsigned pixel;
 x0 = pm->mx[0]; y0 = pm->my[0];
 x1 = pm->mx[pm->nb-1] + textwidth(pm->text[0]);
 y1 = pm->my[pm->nb-1] + dy;
 pixel = imagesize(x0,y0,x1,y1);
 image = (char *)farmalloc(pixel);
 getimage(x0,y0,x1,y1,image);
}
/*********************************************************************/
reset_image(pm)
```

```
POP_MENU *pm;
{
 int x0,y0;

 if (image)
  {
   x0 = pm->mx[0];
   y0 = pm->my[0];
   putimage(x0,y0,image,COPY_PUT);
   farfree(image);
   image = (char *)0;
  }
}

/*********************************************************************/
draw_title()
{
 struct PTS {int x,y;};
 struct PTS pt[]= {0,0,92,104,56,104,56,64,0,48,-56,64,-56,104,-92,104,0,0};
 int i,x,y,x1,y1;
 draw_frame(0,0,639,479,DARKGRAY);
 title("Menu offre ASTRON",0);
 settextstyle(0,0,0);
 setcolor(WHITE);
 x = 479 - textheight("C") - (25-textheight("C"))/2;
 outtextxy(150,x,"(C) Commercial Intertech S.A.  1988-1989-1990");
 for (x = 0;x < 9;x++)
  {
   pt[x].x = pt[x].x*.4 + 45;
   pt[x].y = pt[x].y*.4 + 410;
  }
 setcolor(WHITE);
 setfillstyle(1,LIGHTBLUE);
 fillpoly(9,(int far *)pt);
 settextstyle(2,0,0);
 setusercharsize(9,4,2,1);
 setcolor(WHITE);
 outtextxy(pt[7].x,pt[7].y,"ASTRON");
 settextstyle(0,0,0);
 }
/*********************************************************************/
 cyprion()
{
 struct PTS {int x,y;};
 struct PTS pt[]= {65,5 ,48,4 ,33,5 ,25,6 ,20,7 ,15,9 ,11,11 ,8,14 ,6,16 ,
                   5,19 ,4,23 ,5,27 ,6,30 ,9,33 ,13,37 ,21,41 ,33,45 ,49,47 ,
```

```
                57,47 ,66,47 ,73,44 ,69,39 ,67,35 ,65,31 ,57,34 ,54,35 ,
                48,35 ,44,34 ,40,32 ,37,29 ,34,26 ,34,22 ,36,19 ,40,17 ,
                45,15 ,50,15 ,53,16 ,63,19 ,64,12 ,65,8 ,65,6 ,70,9 ,
                73,12 ,80,17 ,87,23 ,89,26 ,91,30 ,91,34 ,91,38 ,89,41 ,
                86,44 ,123,45 ,117,40 ,115,37 ,113,34 ,112,30 ,113,26,
                115,21 ,117,17 ,122,11 ,130,6 ,138,18 ,140,26 ,140,31,
                140,37 ,138,44 ,172,44 ,169,40 ,167,37 ,166,33 ,172,34 ,
                179,35 ,189,35 ,194,33 ,198,30 ,201,27 ,203,24 ,204,32 ,
                203,39 ,201,44 ,238,44 ,234,40 ,231,37 ,230,33 ,235,34 ,
                238,35 ,246,35 ,249,38 ,252,41 ,253,44 ,307,44 ,303,38 ,
                300,33 ,302,34 ,308,38 ,312,40 ,319,43 ,324,44 ,332,45 ,
                341,45 ,350,44 ,355,42 ,360,39 ,364,36 ,367,32 ,367,35 ,
                367,39 ,365,44 ,394,45 ,392,38 ,391,33 ,408,44 ,436,44 ,
                431,39 ,429,35 ,427,29 ,425,25 ,424,20 ,424,13 ,425,8 ,
                427,4 ,397,4 ,400,15 ,384,4 ,355,4 ,360,11 ,363,17 ,
                360,14 ,352,9 ,346,7 ,341,6 ,333,4 ,326,4 ,322,4 ,313,6 ,
                308,7 ,303,10 ,300,13 ,298,16 ,296,20 ,297,14 ,298,5 ,
                264,4 ,270,14 ,273,24 ,274,28 ,275,31 ,273,43 ,266,35 ,
                264,29 ,266,27 ,267,23 ,266,17 ,261,11 ,257,8 ,247,5 ,
                222,3 ,207,3 ,193,4 ,196,9 ,199,13 ,190,8 ,184,6 ,180,5 ,
                153,3 ,142,3 ,131,5 ,115,5 ,100,19 ,80,5 ,66,5 };

int i,j,dx = 30,dy = 130;
char text[] = "L'outil du Batisseur ASTRON";
char temp;
for (i = 0;i < 171;i++)
  {
   pt[i].x = pt[i].x*1.3+dx;
   pt[i].y = pt[i].y*1.3+dy;
  }
setcolor(WHITE);
draw_frame(0,0,639,479,-WHITE);
moveto(pt[0].x,pt[0].y);
for (i = 1;i < 171;i++)
 {
  lineto(pt[i].x,pt[i].y);
  for (j = 0;j < 200;j++)
    {
     if (kbhit())
        {
         i = getch();
         if (i == 0) getch();
         setcolor(WHITE);
         cleardevice();
         return;
        }
```

```
    }
  }
setfillstyle(SOLID_FILL,MAGENTA);
fillpoly(171,(int far *)pt);

setfillstyle(SOLID_FILL,BLACK);
fillellipse(218+dx,25+dy,14,10);
fillellipse(308+dx,25+dy,14,10);
fillellipse(429+dx,31+dy,16,13);
settextstyle(1,0,0);
setusercharsize(5,4,5,4);
for (i = 1;i <= strlen(text);i++)
  {
   temp = text[i];
   text[i] = '\0';
   outtextxy(50,350,text);
   text[i] = temp;
   for (j = 0;j < 200;j++)
     {
      if (kbhit()) getch();
     }
  }
getch();
setcolor(WHITE);
cleardevice();
}
/*=========================================================================
===*/
get_menus(dy)
int dy;

{
 int i,j,n;
 int nm = 5;
 pm = (POP_MENU *)malloc(nm*sizeof(POP_MENU));
 pm[0].text[0] = " DESCRIPTION PROJET ";
 pm[0].text[1] = "  CHIFFRAGE  ";
 pm[0].text[2] = " RESULTATS  ";
 pm[0].text[3] = " GRAPHIQUES ";
 pm[0].text[4] = "  PROJETS  ";
 pm[0].text[5] = "  RETOUR  ";
 pm[0].text[6] = NULL;
 pm[0].color = BLUE;
 pm[0].colora = BROWN;
 pm[0].mx[0] = 2;
 pm[0].my[0] = 41;
```

```
for (i = 1;i < 6; i ++)
  {
   pm[0].mx[i] = pm[0].mx[i-1] + textwidth(pm[0].text[i-1]) + 2;
   pm[0].my[i] = pm[0].my[i-1];
  }

pm[0].act[0] = "$M1";
pm[0].act[1] = "TRANSFER";
pm[0].act[2] = "$M2";
pm[0].act[3] = "$M3";
pm[0].act[4] = "$M4";
pm[0].act[5] = "EXIT";

pm[1].text[0] = " DONNEES CLIENT     ";
pm[1].text[1] = " BATIMENT & CHARGES ";
pm[1].text[2] = " ARCHITECTURAUX     ";
pm[1].text[3] = " AUTOSTABILITE      ";
pm[1].text[4] = " ACCESSOIRES        ";
pm[1].text[5] = " ISOLATION          ";
pm[1].text[6] = " BARDAGE INTERIEUR  ";
pm[1].text[7] = " SOUS BASEMENTS     ";
pm[1].text[8] =  NULL;
pm[1].color = LIGHTGREEN;
pm[1].colora = LIGHTGRAY;
pm[1].mx[0] = pm[0].mx[0];
pm[1].my[0] = pm[0].my[0] + dy + 2;

pm[1].act[0] = "C:\\TC\\CLIENT";
pm[1].act[1] = "C:\\TC\\MAINBLD";
pm[1].act[2] = "C:\\TC\\WU";
pm[1].act[3] = "C:\\TC\\AUTOSTAB";
pm[1].act[4] = "C:\\TC\\ACCESSOIRES";
pm[1].act[5] = "C:\\TC\\ISOL";
pm[1].act[6] = "C:\\TC\\BARD";
pm[1].act[7] = "C:\\TC\\SOUSBAS";

pm[2].text[0] = " RESUME     ";
pm[2].text[1] = " DETAILLE   ";
pm[2].text[2] = " ISOLATION  ";
pm[2].text[3] = " MONTAGE    ";
pm[2].text[4] = " TRANSPORT  ";
pm[2].text[5] = " REACTIONS  ";
pm[2].text[6] = NULL;
pm[2].color = LIGHTGREEN;
pm[2].colora = LIGHTGRAY;
pm[2].mx[0] = pm[0].mx[2];
```

```
pm[2].my[0] = pm[0].my[2] + dy + 2;

pm[2].act[0] = "...";
pm[2].act[1] = "...";
pm[2].act[2] = "...";
pm[2].act[3] = "...";
pm[2].act[4] = "...";
pm[2].act[5] = "...";

pm[3].text[0] = " ASTRON     ";
pm[3].text[1] = " ARCHITRION ";
pm[3].text[2] = NULL;
pm[3].color = LIGHTGREEN;
pm[3].colora = LIGHTGRAY;
pm[3].mx[0] = pm[0].mx[3];
pm[3].my[0] = pm[0].my[3] + dy + 2;

pm[3].act[0] = "EMU-TEK.BAT";
pm[3].act[1] = "TOARCHI";

pm[4].text[0] = "  LISTE    ";
pm[4].text[1] = "  CHOIX    ";
pm[4].text[2] = "  COPIE    ";
pm[4].text[3] = NULL;
pm[4].color = LIGHTGREEN;
pm[4].colora = LIGHTGRAY;
pm[4].mx[0] = pm[0].mx[4];
pm[4].my[0] = pm[0].my[4] + dy + 2;

pm[4].act[0] = "...";
pm[4].act[1] = "...";

for (i = 0;i < nm;i++)
 {
  n = 0;
  while(1)
   {
    if (pm[i].text[n] == NULL) break;
    n++;
   }
  pm[i].nb = n;
  if (i > 0)
   {
    for (j = 1;j < n;j++)
       {
        pm[i].mx[j] = pm[i].mx[j-1];
```

```
            pm[i].my[j] = pm[i].my[j-1] + dy + 2;
          }
        }
      }
    }
```

```
/*****************************************************************************/
/*  Introduction des donnees du batiment principal et des charges      */
/*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include <dos.h>

#include "\cyprion\color.h"
#include "\cyprion\bldunit.h"      /* contient la structure S_BU */
#include "\cyprion\valid.h"
#include "\cyprion\bayspa.h"       /* contient la structure S_BS */
#include "\cyprion\mouse.h"
#include "\cyprion\datamenu.h"
#include "\cyprion\wu\charges.h"   /* contient la structure S_NV */
#include "\cyprion\coordfsc.h"
#include "\cyprion\fsc_cond.h"
#include "\cyprion\nb_struc.h"
#include "\cyprion\iso_bar.h"      /* contient la structure S_ISOL_BARD */
#include "\cyprion\wu\print_wa.h"
#include "\cyprion\draw_box.c"
#include "\cyprion\cli.h"          /* contient la structure S_CLIENT */

#define ADD    1
#define CHANGE 2
#define AXONO  3
#define RETURN 4
#define ZOOM   5
#define extern unsigned _stklen = 10000

int read_data_add();
int read_data_ch();
int pop_up();
int maj_bp();
int  title();
int get_val_string();
int get_choice();
int chercher();


S_BU      *pbu;
S_CLIENT  *pcli;
```

```
S_NV     *pNV;
S_BS     *pbs;
S_NBSTRUC *pstruc;
S_MENU    *mmenu;
S_MENU    *bdata;
S_MENU    *ln;
S_FSC          *pfsc;
S_FSC_CON *pfsc_con;
S_ISOL_BARD *pisobar;



int xmouse,ymouse,bmouse,action;
int num_ref,k;
int  max_unit ;        /* nombre d'unites de batiments */
int max_nv = 1;        /* nombre de structures de charges */
int  nm;              /* nm : nombre de champs du menu principal */
int bld_unit = 0;      /* numero du batiment principal  */
int nline = 47;        /* nombre de champs dans la fenetre */
int aux = 0;           /* contient l'indice du dernier espacement lu dans  */
                       /* pbs. Utilise a l'affichage des espacements de    */
                       /* travee dans le cas d'une modification.           */
int x = 30, y = 100;   /* coordonnees du coin superieur gauche de la
fenetre*/
int posx,maxx,maxy,posy,dx,dy,num_err,menu,nmax;

int test = 0;      /* pour indiquer la fin d'introduction des          */
                   /* espacements des travees                          */
char tableau[80];  /* contient le string des espacements des travees   */
char espace[12][9]; /* contient les strings des espacements des travees */
                   /* qui seront affiches.                             */
                   /* exemple : "4*5,6*7.5,3*7" */
                   /* exemple: espace[0] = "4*5", espace[1] = "6*7.5",       */
                   /*          espace[2] = "3*7                        */
                   /* En fait, "espace" facilite la manipulation de chaque*/
                   /* string a part. "tableau" est utile a la fonction  */
                   /* qui calcule le vecteur des espacements a partir d'une */
                   /* chaine de caracteres en entree.                  */
main()
{
 int gdrive = DETECT,gmode;
 int key_pad;
 int i,j,n;      /* variables de travail */
 char string[50]; /* contient la chaine correspondent a la valeur d'une
donnee */
 char fname[15];  /* contient le nom d'un fichier */
```

```
float limite;   /* limite superieur ou inferieur de la valeur d'une donnee */
int introduit ; /* introduit = 1 si les donnees du batiment principal */
                /*              sont deja introduites;          */
                /*         = 0 sinon;                           */
int x0,y0,x1,y1;
int color = 7;  /* coueur des messages */
int ancien;     /* numero du dernier champs actif du menu principal  */
int field_box;  /* numero du champs actif de la fenetre              */
int menu_box;   /* numero du champs actif du menu principal          */
int ginp,key,ps;

initgraph(&gdrive,&gmode,"");
cleardevice();
mycolors();
draw_frame(0,0,639,479,BLACK);

espace[0][0] = '\0';

bdata = (S_MENU *)malloc(nline*sizeof(S_MENU));
pfsc  = (S_FSC *)0;

/* lecture dans le fichier de la derniere offre active et du dernier */
/* numero d'offre attribue par le programme. i prend une valeur      */
/*  differente de 1 si le fichier n'existe pas. Dans ce cas, il faut */
/* d'abord introduire la fiche d'un client.                          */
i = load_last("last_ofn.dat",&j,&num_ref) ;
if (i != 1)
  {
    sound_bell();
    message_erreur(31,LIGHTCYAN,0);
    getch();
    return;
  }
/* si le fichier existe, le titre avec le numero de l'offre est affiche */
title("batiment principal",num_ref);
/***initialisation du menu principal ***/
get_main_bp(&nm,&mmenu);

/* avec le numero de l'offre, le nom du fichier des donnees est calcule. */
/* si le fichier existe , la valeur j de la fonction de lecture est      */
/* differente de -3. On teste sur les valeurs de la structure S_NBRSTRUC.*/
/* Cette structure contient le nombre de batiments introduits. Si ce     */
/* est strictement superieur a 0, introduit prend la valeur 1,  sinon    */
/* valeur 0;                                                             */
sprintf(fname,"c%05d.DAT",num_ref);
j = load_f(fname);
```

```
if (j == -3)
 {
  sound_bell();
  print_warning(32,RED,0.0);
  getch();
  return;
 }
if (j != -1 && j != -2 && pstruc[0].nbr_unites != 0 && j != 0 && j != -3)
 {
  introduit = 1;  /* donnees sur batiment principal deja introduites */

  max_unit = pstruc[0].nbr_unites;/* nombre des structures S_BU deja */
  /* introduites. Donc le batiment principal plus le nombre de ces  */
  /* architecturaux.                                                */

  max_nv  = pstruc[0].nbr_ch;  /* nombre de structures de charges*/
  /* ce nombre vaut toujours 1, car les charges de neige et de vent  */
  /* sont communs au batiment pprincipal et a ses annexes;          */

  /* La boucle qui suit, calcule les strings des espacements des    */
  /* travees a partir de la structure S_BS relative au travees du   */
  /* batiment principal. Cette strucure sera utile pour afficher    */
  /* un par un les strings des travees et aussi pour pouvoir        */
  /* facilement selectionner un de ces string lors d'une eventuelle */
  /* modification.                                                  */
  k = 0;
  i = 0;
  while (i < 12 &&  k < pbu[0].nbrtrv )
      {
       espacement(&string,&k,pbu,bld_unit,pbs);
       strncpy(espace[i],string,9);
       i++;
      }
  espace[i][0] = '\0';
  /* le string tableau est calcule en concatenant tous les strings- */
  /* elements du tableau "espace".                                  */
  strcpy(tableau,espace[0]);
  i = 1;
  while (espace[i][0] != '\0')
      {
       strcat(tableau,",");
       strcat(tableau,espace[i]);
       i++;
      }
 }
```

```
/* si les donnees du batiments principal ne sont pas deja introduites: */
else  if (j == -1 || pstruc[0].nbr_unites == 0 || j == 0)
  {
    max_unit = 1;  /* une unite qui est le batiment principal */
    introduit = 0;

    /* allocation de memoire por les differentes structures: */
    pbu  = (S_BU *)malloc(max_unit*sizeof(S_BU));
    pbs  = (S_BS  *)malloc((2*max_unit+1)*sizeof(S_BS));
    pNV  = (S_NV *)malloc(max_nv*sizeof(S_NV));


/***initialisation des structures ***/

    for (j=0;j < max_nv;j++)
         init_nv(j,pNV);
    for (j=0;j < max_unit;j++)
         init_batp(j,pbu);
    for (j=0;j < 2*max_unit+1;j++)
         init_bs(j,pbs);

    pstruc[0].nbr_unites = max_unit;      /* nombre d'unites       */
    pstruc[0].nbr_ch     = max_nv;        /* nombre de charges     */
    pstruc[0].nbr_bs     = 2*max_unit + 1; /* une structure S_BS pour */
    /* les espacements des travees, une pour les colonnes du pigon   */
    /* avant et une pour les colonnes du pignons arriere             */
   }
 /* dessin du batiment principal si deja cree */
 if (introduit) dr_project(max_unit,pbu,pbs,&pfsc,pfsc_con,1);

/***dessin du menu principal***/
 init_drawbox(0,0);
 for (j = 1 ; j < nm ;j++)
   draw_box(j,mmenu,cmen,cmtx,mmenu[j].text);
/*** initialisation de la souris***/
 hello_mickey();
 set_mouse_sensitivity(4,8);
 mouse_display();

/*** initalisation des variables***/
 menu_box  = -1;
 field_box = -1;
 bmouse    = 0;
 action    = 0;
 ancien    = -1;
 ginp = 0;
```

```
 maxx = 0;
 maxy = 0;

while(1)
 {
  kbgetkeymouse(&ps,&key,&bmouse,&xmouse,&ymouse);

/* si le bouton de la souris n'est pas relache */
  if (bmouse != 0)
    {
     if ((menu_box = box_number(xmouse,ymouse,mmenu,nm,0,0)) != -1)
      {
        if (action == 2)
            {
               mouse_hide();
               aux = 0;
               dr_project(max_unit,pbu,pbs,&pfsc,pfsc_con,1);
               mouse_display();
               ginp = 0;
               action = 0;
            }
        if (menu_box !=  ancien)
         {
          mouse_hide();
          init_drawbox(0,0);
         draw_box(menu_box,mmenu,cmbox,cmtx,mmenu[menu_box].text);
          if (ancien != -1)
              draw_box(ancien,mmenu,cmen,cmtx,mmenu[ancien].text);
          ancien = menu_box;
          init_drawbox(x,y);
          mouse_display();
         }
      }
        else if (action == CHANGE && ginp == 1)
            {
              /* saisie du numero du champ de la donnee a modifier */
              if ((field_box = box_number(xmouse,ymouse,bdata,nline,x,y))
!= -1)
                {
                  mouse_hide();
                  /* les valeurs de la hauteur libre(champs 35),       */
                  /* des espacements des colonnes du pigon arriere (42) */
                  /* et ceux du pigon avant (40) ne peuvent etre modifies */
                  if (field_box != 35 && field_box != 40 && field_box != 42)
                   {
                      encode_value(string,field_box,0);
```

```
                draw_box(field_box,bdata,cwactbox,cmtx,string);
                draw_message(field_box,bdata);
              }
             ginp = 2;
            }
          }
      else if (action == ZOOM && !ginp)
          {
           ginp = 1;
           x0 = xmouse - 1;
           y0 = ymouse - 1;
          }
       else if (action == ZOOM && ginp == 2)
          {
           ginp = 3;
           x1 = xmouse - 1;
           y1 = ymouse - 1;
          }
       }

else if (!bmouse)  /* si le bouton est relache  */
      {
       if (menu_box !=-1) /* si un champs du menu principal est active*/
        {
          if (strcmp(mmenu[ancien].fmt,"ADD") == 0)
           {

            /* la creation d'un batiment principal n'est possible */
            /* que si le batiment n'est pas deja introduit; Donc, */
            /* introduit == 0.                                    */
            if (introduit == 0)
                      action = ADD;
            else
                {
                 sound_bell();
                 message_erreur(22,7,0.0);
                 action = 0;
                 menu_box = -1;
                }
           }
          else if (strcmp(mmenu[ancien].fmt,"CHANGE") == 0)
              {
                /* la modification des donnees du batiment principal*/
                /* est possible s'il est daja introduit            */
                if (introduit == 1)
                   action = CHANGE;
```

```
                else
                   {
                      sound_bell();
                      message_erreur(20,7,0.0);
                      action = 0;
                      menu_box = -1;
                      ginp = 0;
                   }
              }
          else if (strcmp(mmenu[ancien].fmt,"END") == 0)
            action = RETURN;
          else if (strcmp(mmenu[ancien].fmt,"ZOOM") == 0)
            action = ZOOM;
          else if (strcmp(mmenu[ancien].fmt,"AXONO") == 0)
            action = AXONO;
          if (action == ADD || action == CHANGE
            || action == ZOOM || action == RETURN)
                menu_box = -1;
      }
   else if (action == ADD)
      {
         if (introduit == 0)
           {
             mouse_hide();
             get_window_pr(bdata,nline);
             window_size(x,y,nline,bdata,&maxx,&maxy);
             draw_menu(x,y,nline,bdata,maxx,maxy,0);
             mouse_display();
             if (read_data_add(x,y,nline,bld_unit,pbu,bdata) != -1)
                {
                 /* si les donnees sont introduites sans erreurs : */
                 introduit = 1;
                 mouse_hide();
                 dr_project(max_unit,pbu,pbs,&pfsc,pfsc_con,1);
                }
             else
                {
                 /* si erreur d'introduction des donnees ou si     */
                 /* interruption de l'intriduction par la touche   */
                 /* ESC, toutes les valeurs des champs deja traitees*/
                 /* sont perdues et la fenetre est efface.         */
                 setfillstyle(1,cscr);
                 bar(30,60,610,440);
                 action = 0;
                }
             mouse_display();
```

```
            ginp = 0;
        }
    }


else if (action == CHANGE && field_box == -1 && ginp == 0)
    {
      /* affichage de la fenetre pour modification des donnees */
      mouse_hide();
      get_window_pr(bdata,nline);
      window_size(x,y,nline,bdata,&maxx,&maxy);
      draw_menu(x,y,nline,bdata,maxx,maxy,0);
      mouse_display();
      ginp = 1;

    }
else if (action == CHANGE && field_box != -1  && ginp == 2)
    {
      read_data_ch(field_box,x,y,bld_unit,pbu,bdata,&string);
      /* mise a jours des donnees si necessaire */
      maj_bp(field_box,string,bld_unit);
      aux = 0;
      field_box = -1;
      ginp = 1;
    }

else if (action == ZOOM && ginp == 1)
    {
      putpixel(x0,y0,WHITE);
      ginp = 2;
    }
else if (action == ZOOM && ginp == 3)
    {
      setcolor(GREEN);
      mouse_hide();
      line(x0,y0,x1,y1);
      line(x1,y0,x1,y1);
      line(x1,y1,x0,y0);
      line(x0,y1,x0,y0);
      ginp = 0;
    }
else if (action == AXONO)
    {
      dr_project(1,pbu,pbs,&pfsc,pfsc_con,1);
      ginp =0;
    }
```

```
        else if (action == RETURN)
             {

               /***sauvegarde des donnees ***/
              if (introduit == 1)
               {
                 pstruc[0].nbr_unites = max_unit;
                 pstruc[0].nbr_bs = 2*max_unit + 1;
                 pstruc[0].nbr_ch = max_nv;
               }
              else
               {
                 pstruc[0].nbr_unites = 0;
                 pstruc[0].nbr_bs = 0;
                 pstruc[0].nbr_ch = 0;
               }
              sauver_f(fname);
              free(pcli);
              free(pbu);
              free(pNV);
              free(bdata);
              free(pbs);
              if (pstruc->nbr_autos > 0) free(pfsc_con);
              if (pstruc->nbr_isobard > 0)  free(pisobar);
              free(pstruc);
            restorecrtmode();
              exit(0);
             }
        }
   }
}


/*****************************************************************************/
/** fonction intermediaire de travail **/
espacement(string,i,pbu,bld_unit,pbs)
char *string;
int *i;
S_BU  *pbu;
int bld_unit;
S_BS  *pbs;

{
  esp_trv(&string,i,pbu,bld_unit,pbs);
}
/*****************************************************************************/
```

```
/*      Gestion de l'introduction des donnees lors de la creation  d'un */
/*  batiment principal.                                              */
/*  Tous les champs de la fenetre sont parcourus consecutivement.    */
/* principe d'introduction des espacements des travees : les espacements */
/* sont introduit string par string. Chaque string est valide,affecte a  */
/* l'element correpondant de "espace", et concatenne avec le string      */
/* "tableau". La fin d'introduction des espacements est indique par
RETURN*/
/* et alors le string "tableau" est valide.                         */
/* exemple :                                                        */
/*    1er string: 2*5 --> valider --> espace[0] = 2*5 -->tableau = 2*5   */
/*    2er      : 3*9 --> valider --> espace[1] = 3*9 --> 2*5,3*9        */
/*    3er      : 4*5.5             espace[2] = 4*5.5   2*5,3*9,4*5.5   */
/*    4er      :RETURN             espace[3] = '\0'    2*5,3*9,4*5.5   */
/*    la variable test est mise a 1 pour indiquer la fin d'introduction  */
/*    et "tableau" est valide.                                         */
/*********************************************************************/
int read_data_add(x,y,nline,bld_unit,pbu,ln)
int  x,y;     /* pour le positionnement des champs de la fenetre */
int bld_unit;
S_BU   *pbu;
S_MENU  *ln;  /* pour parcourir la fenetre  */
{
 int num_err,i,n;
 int keypad = 0;
 int ok,color=1;

 float limite = 0;
 int j = bld_unit;
 char string[50],str[50];
 int it = 1;       /* numero du champs actif */

 tableau[0] = '\0';

 ln[40].menu = 0;  /* mis a 0 pour l'affichage seulement car c'est */
 ln[42].menu = 0;  /* donnees calculees par le programme et non pas*/
                   /* introduites. */
 while (it < nline)
    {
     while (it < nline  && ln[it].nmax <= 0)  (it++;)
     if (it == nline) break;
     str[0] = '\0';
     test = 0;
     encode_value(string,it,j);
     /* dans le cas des espacements de travees, les strings respectifs */
     /* sont affiches un a un. */
```

```
    if (it >= 25 &&  it <= 34 && string[0] == '\b')
              draw_box(it,ln,cwactbox,cwtx,espace[it-25]);
    else  draw_box(it,ln,cwactbox,cwtx,string);
    draw_message(it,ln);
    if (ln[it].menu == 0)
      {
       keypad = 0;
       if (it >= 25 && it <= 34)
         {
           n =
get_value_esp(ln[it].posdx+x,ln[it].posdy+y,keypad,string,ln[it].nmax,1);
           if (string[0] == '\0')
              test = 1; /* pour signaler la fin de l'introduction  */
                        /* des strings des espacements des travees */
                        /* et passer a un autre donnee */
         }
       else if (it == 46)
           n =
get_value_esp(ln[it].posdx+x,ln[it].posdy+y,keypad,string,ln[it].nmax,2);
       else if (it != 40 && it != 42)
           n =
get_value_pad(maxx+10,y+10,ln[it].posdx+x,ln[it].posdy+y,keypad,string,ln[it
].nmax,1);
       if (n == -1) return(-1);
      if (string[0] == '\b')
        {
           message_erreur(0,7,0.0);
           str[0] = '\b';
           str[1] = '\0';
           if (it >= 25 && it <= 34 ) /* espacement des travees */
            draw_box(it,ln,cwbox,cwtx,espace[it -25]);
           else
            draw_box(it,ln,cwbox,cwtx,string);
           if (it == 25 || it == 23)  it-- ;
           it = it == 1 ? it : it - 1;
           while (ln[it].nmax <= 0 && it > 1) it--;
         }
       if (it >= 25 && it <= 34 && str[0] == '\b')
            draw_box(it,ln,cwactbox,cwtx,espace[it - 25]);
       else
            draw_box(it,ln,cwactbox,cwtx,string);
       /* si introduction d'un string des espacements des travees: */
       if (it >= 25 && it <= 34)
        {
         if (string[0] != '\0')
           {
```

```
            {
             /* les champs des espacements des travees non utilises   */
             /* sont remplis par un string nul, ainsi que les elements */
             /* de "tableau" qui leurs correspondent */
             espace[it-25][0] = '\0';
             while (it <= 34)
              {
               draw_box(it,ln,cwbox,cwtx," ");
               it++;
              }
            }
           if (it == 35)
            {
           encode_value(string,it,j);
              draw_box(it,ln,cwbox,cwtx,string);
              it++;
            }
       /* si pas de back space, passer au champ suivant de la fenetre */
        if (str[0] != '\b')   it++;
           if (it == 40 || it == 42)
            {
             /* dans le cas des espacements des pignons, pas de saisie */
             /* de string a l'ecran */
             encode_value(string,it,j);
             draw_box(it,ln,cwactbox,cwtx,string);
             draw_message(it,ln);
             decode_value(string,it,bld_unit);
             it++;
            }
           if (str[0] != '\b') strcpy(string," ");
        }
       else  if (!ok)
        {
         /* si le string n'est pas valide : */
         sound_bell();
         message_erreur(num_err,7,limite);
         if (it >= 25 && it <= 34 && test == 0)
           { /*tester si test = 0, car sinon le string est nul */
             /* cas des espacements des travees */
             n = strlen(tableau) - strlen(string);
             if (n > 0) tableau[n-1] = '\0';
             else tableau[n] = '\0';
           }
        }
     }
else if (ln[it].menu == 1)
```

```
        {
          if (pop_up(&it,0,x,y,pbu,1) == -1) return(-1);
          it++;
        }
    }

 ln[40].menu = -1; /* les espacement des colonnes des pigons sont */
 ln[42].menu = -1; /* calcules par le programme                   */
 mouse_display();
}
/*****************************************************************************/
/*    Gestion de la modification des valeurs des donnes.          */
/*  Cette fonction ne traite que la donnee correspondant au champ     */
/*  selectionne.
/*****************************************************************************/
int read_data_ch(item,x,y,bld_unit,pbu,ln,string)
int item,x,y,bld_unit;
char *string;
S_BU *pbu;
S_MENU *ln;
{
 int color,n,len,ok,k,num_mess;
 float limite;
 if (item == 35 || item == 40 || item == 42)
  {
    /* les espacements des colonnes des pignons et de la hauteur   */
    /* libre ne peuvent etre modifies.                             */
    sound_bell();
    message_erreur(29,7,0.0);
  }

 if (ln[item].menu == 1)
  {
    pop_up(&item,bld_unit,x,y,pbu,2);
  }
 else if (ln[item].menu == 0)
  {
    while (item)
      {
        if (item >= 25 && item <= 34)
          {
            /* la boucle for calcul le nombre des strings des espacements*/
            /* des travee. */

            for (k =0 ; espace[k][0] != '\0' ;k++);
```

```
            /* les modifications possibles sur les espacements des travees*/
            /* sont : - modification d'un string deja introduit;        */
            /*        - suppression d'un string dejaintroduit;          */
            /*        - ajout d'un string dans le champ qui suit le     */
            /*          le dernier champ non vide et seulement dans     */
            /*          celui-ci.                                       */

            if ( item - 25 > k )
              {
               sound_bell();
               message_erreur(23,7,0.0);
               draw_box(item,ln,cwbox,cwtx," ");
               mouse_display();
               return;
              }
           }
          string[0] = '\0';
          if (item < 25 || item > 34)
             encode_value(string,item,bld_unit);
          draw_box(item,ln,cwactbox,cwtx,string);
          draw_message(item,ln);
          if (item >= 25 && item <= 34)
               n =
get_value_esp(ln[item].posdx+x,ln[item].posdy+y,0,string,ln[item].nmax,1);
          else if (item == 46)
               n =
get_value_esp(ln[item].posdx+x,ln[item].posdy+y,0,string,ln[item].nmax,2);
          else
           {
            n =
get_value_pad(maxx+10,y+10,ln[item].posdx+x,ln[item].posdy+y,0,string,ln[it
em].nmax,2);
           }
          if (n == -1) return(-1);
            mouse_hide();
          draw_box(item,ln,cwactbox,cwtx,string);
          mouse_display();
          decode_value(string,item,bld_unit);
          ok = validate_bp(string,item,&num_mess,&color,&limite);
          if (ok)
           {
            message_erreur(0,0,0.0);
            if (item < 25 || item > 34)
               encode_value(string,item,bld_unit);
            mouse_hide();
            draw_box(item,ln,cwbox,cwtx,string);
```

```
            mouse_display();
            if (item == 21 || item == 23)
              {
               item++;
               encode_value(string,item,bld_unit);
               draw_box(item,ln,cwbox,cwtx,string);
              }
            else if (item == 22 || item == 24)
              {
               encode_value(string,item-1,bld_unit);
               draw_box(item-1,ln,cwbox,cwtx,string);
              }
            item = 0;
           }
          else
           {
            sound_bell();
            message_erreur(num_mess,7,limite);
           }
        }
    }
  mouse_display();
}
/*************************************************************************/
/*  mise a jours des autres donnees apres une modification.        */
/*************************************************************************/
int maj_bp(item,string,bld_unit)
int item;     /* numero du champ de la donnee modifiee */
int bld_unit;
char *string;
{
int ok,nvalue,i,num_mess,k,color;
float limite;
char str[80];
float *value;
  switch(item)
   {
    case 1 : /* si le type du batiment est modifie: */
             /* mise a jours du span */
           sprintf(string,bdata[item].fmt,pbu[bld_unit].span);
           item = 20;
           ok = validate_bp(string,item,&num_mess,&color,&limite);
           if (ok == 0)
            {
              read_data_ch(item,x,y,0,pbu,bdata,string);
              maj_bp(item,string,bld_unit);
```

```
        }

        /* mise a jours des pentes */
       sprintf(string,bdata[item].fmt,pbu[bld_unit].sll);
       item = 21;
       ok = validate_bp(string,item,&num_mess,&color,&limite);
       if (ok == 0)  read_data_ch(item,x,y,0,pbu,bdata,string);

        /* mise a jour des hauteurs aux gouttieres */
       sprintf(string,bdata[item].fmt,pbu[bld_unit].ehl);
       item = 23;
       ok = validate_bp(string,item,&num_mess,&color,&limite);
       if (ok == 0)  read_data_ch(item,x,y,0,pbu,bdata,string);

        /* mise a jours de la hauteur libre */
       item = 35;
       encode_value(string,item,0);
       draw_box(item,bdata,cwbox,cwtx,string);

       break;
case 2 : /* si le tupe de bardage est modifie */
       item++;
       strcpy(string,pbu[bld_unit].wexpfin);
       read_data_ch(item,x,y,0,pbu,bdata,string);

       /* mise a jours des espacements des colonnes du pignon avant */
       item = 40;
       encode_value(string,item,0);
       draw_box(item,bdata,cwactbox,cwtx,string);
       decode_value(string,item,bld_unit);
       draw_box(item,bdata,cwbox,cwtx,string);

       /* mise ajours des espacements des colonnes du pignons arriere */
       item = 42;
       encode_value(string,item,0);
       draw_box(item,bdata,cwactbox,cwtx,string);
       decode_value(string,item,bld_unit);
       draw_box(item,bdata,cwbox,cwtx,string);
       break;

case 4 : /* si le type de couverture est modifie */
       if (strcmp(pbu[bld_unit].rptyp,"DSR") == 0)
         for ( i=5 ; i<= 8 ; i++)
              {
               mouse_hide();
               read_data_ch(i,x,y,0,pbu,bdata,string);
```

```
                mouse_display();
               }
        else
            {
              mouse_hide();
              read_data_ch(7,x,y,0,pbu,bdata,string);
              mouse_display();
              break;
            }


case 5 :

case 6 :
case 8 :
      /* si la couverture exterieur n'est pas de type DSR, alors pas */
      /* de couverture interieuret pas de modification possible pour */
      /* derniere */
      if (strcmp(pbu[bld_unit].rptyp,"DSR") != 0)
          message_erreur(21,7,0.0);
      break;

case 14: /* si introduction ou suppression de la colonne centrale: */
        /* mise a jours de la distribution des espacements des   */
        /* colonnes du pignon avant */
      item = 40;
      encode_value(string,item,0);
      draw_box(item,bdata,cwactbox,cwtx,string);
      decode_value(string,item,bld_unit);
      draw_box(item,bdata,cwbox,cwtx,string);
      break;
case 16: /* idem que pour pignon avant */
      item = 42;
      encode_value(string,item,0);
      draw_box(item,bdata,cwactbox,cwtx,string);
      decode_value(string,item,bld_unit);
      draw_box(item,bdata,cwbox,cwtx,string);
      break;

case 17: /* si modification de la charge de neige ou */
case 18: /* de la charge de vent ou                  */
case 19: /* de la charge additionnelle, alors :      */
        /* recalcul de la hauteur libre            */
      item = 35;
      encode_value(string,item,0);
      draw_box(item,bdata,cwbox,cwtx,string);
```

```
        break;

    case 20:/* modification du span */
          /* mise a jours de la hauteur libre */
        item = 35;
         encode_value(string,item,0);
         draw_box(item,bdata,cwbox,cwtx,string);

          /* mise a jours de la distribution des colonnes du pignons avant
*/
         item = 40;
         encode_value(string,item,0);
         draw_box(item,bdata,cwactbox,cwtx,string);
         decode_value(string,item,bld_unit);
         draw_box(item,bdata,cwbox,cwtx,string);

         /* idem que pignon avant */
         item = 42;
         encode_value(string,item,0);
         draw_box(item,bdata,cwactbox,cwtx,string);
         decode_value(string,item,bld_unit);
         draw_box(item,bdata,cwbox,cwtx,string);
         break;

    case 21 : /* modification de la pente gauche */
             /* mise a jours de la pente droite */
        pbu[bld_unit].slr = pbu[bld_unit].sll;

             /* mise a jours de la hauteur libre  */
        item = 35;
        encode_value(string,item,0);
        draw_box(item,bdata,cwbox,cwtx,string);
        break;

    case 22: /* modification de la pente droite */
           /* mise a jours de la hauteur libre */
       item = 35;
        encode_value(string,item,0);
        draw_box(item,bdata,cwbox,cwtx,string);
        break;

    case 23 : /* modification de la hauteur a la gouttiere gauche */
       pbu[bld_unit].ehr = pbu[bld_unit].ehl;

       /* mise a jours de la hauteur libre */
       item = 35;
```

```
    encode_value(string,item,0);
    draw_box(item,bdata,cwbox,cwtx,string);
    break;

case 24:
    /* idem que precedement */
  item = 35;
    encode_value(string,item,0);
    draw_box(item,bdata,cwbox,cwtx,string);
    break;

case 25:
case 26:
case 27:
case 28:
case 29:
case 30:
case 31:
case 32:
case 33:
case 34:
    /* recalcul de la distribution des espacements des travees */
    /* et des strings a afficher */

    /* calcul de l'indice du string dans le tableau espace.     */
    /* Les champs contenant les espacements des travees sont les */
    /* champs 25 jusqu'a 35.                                    */
    i = (item - 25);
    /* calcul de la dimension de "espace" et des champs pouvant  */
    /* etre modifies.                                           */
    for (k=0 ; espace[k][0] != '\0'; k++);
  if (k >= i)
     {
      /* si le string est non nul: */
      if (string[0] != '\0')
       {
          strcpy(espace[i],string);
          if (i == k)
                espace[i+1][0] = '\0';
       }
      else  /* string nul = suppression d'un string deja introduit. */
            /* Les elements du tableau espace a partir de l'element */
            /* msupprime sont avances d'un indice. La nouvelle     */
            /* distribution est affichee. */
       {
           mouse_hide();
```

```
                for (k = i ; espace[k][0] != '\0' ; k++)
                  {
                     strncpy(espace[k],espace[k+1],9);
                     draw_box(k+25,bdata,cwbox,cwtx,espace[k]);
                  }
                 mouse_display();
                 espace[k][0] = '\0';
               }

            /* calcul de la nouvelle valeur de la variable tableau */
            strcpy(str,espace[0]);
            i = 1;
            while (espace[i][0] != '\0')
                 {
                   strcat(str,",");
                   strcat(str,espace[i]);
                   i++;
                 }
            strcpy(tableau,str);
            /* transformation du string tableau en vecteur de floats */
            get_val_string(str,&nvalue,&value);

            /* mise a jours du tableau contenant le vacteur des espacements
*/
            /* des travees.(la structure S_BS). */
            for (i = 0; i < nvalue; i++)
               pbs[pbu[0].bssw].bs[i] = value[i];
            for (i = nvalue; i < 30 ;i++)
               pbs[pbu[0].bssw].bs[i] = 0.0;
            pbu[0].nbrtrv = nvalue;
            pbu[0].tobay = pbu[0].nbrtrv;
          }

         /* mise a jours de la hauteur libre */
        item = 35;
         encode_value(string,item,0);
         draw_box(item,bdata,cwbox,cwtx,string);

         break;

    default:
       item = 0;
    }
}
```

```
/*************************************************************************/
/*          Fonction qui gere les pop-up menus                           */
/*************************************************************************/
int pop_up(item,bld_unit,x,y,pbu,act)
int *item,bld_unit,x,y,act;
S_BU *pbu;

#define BXO bdata[*item].bx + bdata[*item].dx + x
#define BYO bdata[*item].by + y
{
char *choice,*null_choice = "\0";
int i,ok,erreur;
char string[50];
int num_err;
float limite;
int color = 1;

 switch(*item)
 {
  case 1 : /*type du batiment principal*/
        /* bldtyppr = liste des types possibles pour le batiment principal*/
        erreur = get_choice(BXO,BYO,bldtyppr,&choice,0,act);
        break;

  case 2 : /*type du bardage*/
        erreur = get_choice(BXO,BYO,wptyp,&choice,0,act);
        break;

  case 3 : /*finition pour le type de bardage choisi */
        i = compar(pbu[bld_unit].wexptyp,wptyp) ;
        if (i != -1)
        erreur = get_choice(BXO,BYO,wpfin[i],&choice,0,act);
      break;

  case 4 :  /*couverture*/
        erreur = get_choice(BXO,BYO,rptyp,&choice,0,act);
        break;

  case 5:
      if (strcmp(pbu[bld_unit].rptyp,"DSR") == 0)
         erreur = get_choice(BXO,BYO,rptyp,&choice,2,act);
      else choice = null_choice;
      break;
```

```
case 6 :
    if (strcmp(pbu[bld_unit].rptyp,"DSR") == 0)
      erreur = get_choice(BXO,BYO,rptyp,&choice,2,act);
    else choice = null_choice;
    break;

case 7 :
    if (strcmp(pbu[bld_unit].rptyp,"DSR") == 0)
     {
      if (( i= compar(pbu[bld_unit].rexptyp,rptyp)) != -1)
       erreur = get_choice(BXO,BYO,rpfin[i],&choice,0,act);
     }
    else if ((i= compar(pbu[bld_unit].rptyp,rptyp)) != -1)
         erreur = get_choice(BXO,BYO,rpfin[i],&choice,0,act);
    break;

case 8 :
    if (strcmp(pbu[bld_unit].rptyp,"DSR") == 0)
     {
      if ((i= compar(pbu[bld_unit].rinptyp,rptyp)) != -1)
       erreur = get_choice(BXO,BYO,rpfin[i],&choice,0,act);
     }
    else  choice = null_choice;
    break;

case 9 : /*finition ossature primaire*/
    erreur = get_choice(BXO,BYO,primfin,&choice,0,act);
    break;

case 10 :/*raccord descente d'eau*/
    erreur = get_choice(BXO,BYO,delwallcom,&choice,0,act);
    break;

case 11 :/*type de gouttiere*/
    erreur = get_choice(BXO,BYO,gutter,&choice,0,act);
    break;

case 12 :/*type de gouttiere*/
    erreur = get_choice(BXO,BYO,gutter,&choice,0,act);
    break;

case 13 : /*type pignon avant */
    if ((i = compar(pbu[bld_unit].bldtyp,bldtyppr)) != -1)
     erreur = get_choice(BXO,BYO,ewtyp,&choice,0,act);
    break;
```

```
    case 14 :/*colonne central sur pigon avant*/
         erreur = get_choice(BXO,BYO,delwallcom,&choice,0,act);
         break;

    case 15 : /*type pigon arriere*/
         if ((i = compar(pbu[bld_unit].bldtyp,bldtyppr)) != -1)
          erreur = get_choice(BXO,BYO,ewtyp,&choice,0,act);
         break;

    case 16 :/*colonne central sur pignon arriere*/
         erreur = get_choice(BXO,BYO,delwallcom,&choice,0,act);
         break;

   }
   if (erreur == -1) return(-1);
   decode_value(choice,*item,bld_unit);
   encode_value(string,*item,bld_unit);
   init_drawbox(x,y);
   if (*item == 4 && (strcmp(pbu[bld_unit].rptyp,"DSR") != 0))
    {
     draw_box(*item,bdata,cwbox,cwtx,string);
     draw_box((*item)+1,bdata,cwbox,cwtx,"\0");
     strcpy(pbu[0].rinptyp," ");
     draw_box((*item)+2,bdata,cwbox,cwtx,"\0");
     strcpy(pbu[0].rexptyp," ");
    }
   else
     draw_box(*item,bdata,cwbox,cwtx,string);
   ok = validate_bp(choice,*item,&num_err,&color,&limite) ;
   if (ok != 0)
    {
/*    sound_bell();*/
     message_erreur(num_err,color,limite);
    }
   return(1);
}
/*************************************************************************/
/*              messages d'erreurs                                       */
/*************************************************************************/
message_erreur(num_err,color,limite)
int color;
int num_err;
float limite;
{
 int bkcol,frcol;
 char buffer[60];
```

```
bkcol = getpixel(1,430);
frcol = getcolor();
setfillstyle(1,bkcol);
bar(1,430,638,450);
if (!color) return;
setcolor(color);
switch(num_err)
{
case 1 :
    outtextxy(110,435,"cette valeur est obligatoire");
    break;

case 2 :
   sprintf(buffer," les espacements doivent etre plus petits que
%3.0f",limite);
   outtextxy(110,435,buffer);
   break;

case 3 :
    sprintf(buffer,"la portee doit etre inferieur ou egal a %7.3f",limite);
    outtextxy(110,435,buffer);
    break;

case 4 :
    sprintf(buffer,"la portee doit etre superieure ou egale a %7.3f",limite);
    outtextxy(110,435,buffer);
    break;

case 5 :
    sprintf(buffer,"la pente doit etre superieure ou egale a %6.2f",limite);
    outtextxy(110,435,buffer);
    break;

case 6 :
    sprintf(buffer,"la pente doit etre inferieure ou egale a %6.2f",limite);
    outtextxy(110,435,buffer);
    break;

 case 7 :
    sprintf(buffer,"hauteur a la gouttiere superieure ou egal %6.2f",limite);
    outtextxy(110,435,buffer);
    break;

 case 8 :
    sprintf(buffer,"hauteur a la gouttiere inferieure ou egal %6.2f",limite);
    outtextxy(110,435,buffer);
```

```
      break;

   case 9 :
      sprintf(buffer," nombre de travees inferieur ou egal %2f",limite);
      outtextxy(110,435,buffer);
      break;

   case 10 :
      outtextxy(110,435,"la pente gauche doit etre positive");
      break;

   case 11 :
      outtextxy(110,435,"la portee doit etre positive");
      break;

   case 12 :
      outtextxy(110,435,"la portee doit etre un multiple de 400 ");
      break;

   case 13 :
      sprintf(buffer,"la charge de neige maximum est de %7.3f",limite);
      outtextxy(110,435,buffer);
      break;

   case 14 :
      sprintf(buffer,"la charge de vent maximum est de %7.3f",limite);
      outtextxy(110,435,buffer);
      break;

   case 15 :
        sprintf(buffer,"le nombre de travees minimum est %f",limite);
        outtextxy(110,435,buffer);
        break;

    case 16 :
        sprintf(buffer,"limite maximum pour la permeabilite est
%6.2f",limite);
        outtextxy(110,435,buffer);
        break;

    case 17 :
        outtextxy(110,435,"le batiment principal ne peut etre supprimer");
        break;

    case 18 :
        outtextxy(110,435,"cette valeur doit etre non nulle");
```

```
        break;

    case 19 :
        outtextxy(110,435,"cette valeur doit etre positive");
        break;

    case 20:
        outtextxy(110,435,"veuillez introduire d'abord les donnees du
batiment principal");
        break;

    case 21:
        outtextxy(110,435,"la couverture doit etre de type DSR");
        break;

    case 22:
         outtextxy(110,435,"le batiment principal est deja introduit");
         break;

    case 23:
         outtextxy(110,435,"seules les cases des donnees ou celle qui suit
sont permises");
         break;

    case 24:
         outtextxy(110,435,"les espacements sont obligatoires");
         break;

    case 25:
         outtextxy(110,435,"colonne centrale obligatoire pour cette portee");
         break;

    case 26:
         outtextxy(110.435,"colonne centrale obligatoire pour ce type de
batiment");
         break;

    case 27:
         sprintf(buffer,"la limite superieur est %7.3f",limite);
         outtextxy(110,435,buffer);
         break;

    case 28:
         outtextxy(110,435,"cette valeur doit etre positive");
         break;
```

```
    case 29:
        outtextxy(110,435," Cette valeur ne peut etre modifiee ");
        break;

    case 30:
        sprintf(buffer,"l'espacement doit etre superieur ou egale a
%7.3f",limite);
        outtextxy(110,435,buffer);
        break;

    case 31 :
        outtextxy(115,200,"VEUILLEZ INTRODUIRE D'ABORD LA FICHE
CLIENTD ");
        outtextxy(115,220,"APPUYEZ SUR UNE TOUCHE POUR CONTINUER
SVP:");
        break;

    case 32 :
        outtextxy(115,200,"VEUILLEZ VERIFIER L'EXISTENCE DU FICHIER ");
        outtextxy(115,220,"APPUYEZ SUR UNE TOUCHE POUR CONTINUER
SVP:");
        break;

    case 33:
        sprintf(buffer,"la surface par descente d'eau doit etre inferieur ou
e2gale a %7.3f",limite);
        outtextxy(110,435,buffer);
        break;

    case 34:
        sprintf(buffer,"numero d'axe entre 2 et %f",limite);
        outtextxy(110,435,buffer);
        break;

    case 35:
        outtextxy(110,435,"le premier axe ne peut etre un flanc parallele");
        break;

    case 36:
      outtextxy(110,435,"le dernier axe ne peut etre un flanc parallele");


    default :
      outtextxy(110,435," ");

  }
```

```
  setcolor(frcol);
 }
/*****************************************************************************/
/*        recherche de l'indice d'un type de batiment                        */
/*****************************************************************************/
int chercher(ind,type,liste)
int *ind;
char type[5];
char *liste[];
{
  *ind = 0;
  while ((liste[*ind] != NULL)  && (strcmp(liste[*ind],type) !=0))  (*ind)++ ;
  if (liste[*ind] == NULL) return(-1);
  else return(*ind);
 }
```

```
/*****************************************************************************/
/* Fonctions d'introduction des fiches des clients.                          */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include <dos.h>

#include "\cyprion\color.h"
#include "\cyprion\mouse.h"
#include "\cyprion\wu\charges.h"
#include "\cyprion\bayspa.h"
#include "\cyprion\datamenu.h"
#include "\cyprion\cli.h"
#include "\cyprion\init.h"
#include "\cyprion\nb_struc.h"
#include "\cyprion\bldunit.h"
#include "\cyprion\draw_box.c"
#include "\cyprion\wu\print_wa.h"
#include "\cyprion\coordfsc.h"
#include "\cyprion\fsc_cond.h"
#include "\cyprion\iso_bar.h"

#define EDITION   2   /* edition d'une fiche client pour modification */
#define CREATION  1   /* creation d'une nouvelle fiche client        */
#define SELECTION 3   /* selection d'une fiche deja cree             */
#define COPIE     4   /* copie de la fiche active dans une nouvelle fiche */
#define DESSIN    5   /* dessin de l'offre active                    */
#define RETURN    6
#define extern unsigned _stklen = 10000

int mycolors();
int read_data_cli();
int pop_up();
int title();

S_CLIENT  *pcli;
S_MENU    *ln;
S_MENU    *bdata;
S_MENU    *mmenu;
S_BU      *pbu;
```

```
S_BS     *pbs;
S_NV     *pNV;
S_ISOL_BARD *pisobar;
S_NBSTRUC *pstruc;
S_FSC    *pfsc;
S_FSC_CON *pfsc_con;

int xmouse,ymouse,bmouse,ps,key,action;
int max_struct = 1,num_struct = 0;
int last_ofnbr;        /* numero de la derniere offre active */
int ofnbr;             /* dernier numero d'offre attribue    */
int nm;
int nline = 14;        /* nombre de champs dans la fenetre   */
int x = 50 , y = 100;
int posx,posy,maxx,maxy,dx,dy,menu,nmax;
int existe;

main()
{
 int gdrive = DETECT,gmode;
 int key_pad;
 int i,j,num,n;
 int x0,y0,x1,y1;
 int ancien,menu_box,ginp,field_box;
 char fname[15];
 char string[50];
 initgraph(&gdrive,&gmode,"");
 cleardevice();
 mycolors();
 draw_frame(0,0,639,479,BLACK);


 bdata = (S_MENU   *)malloc(nline*sizeof(S_MENU));
/** lecture de la derniere offre active          **/
/**  et initialisation de la structure pcli      **/

 i = (load_last("last_ofn.dat",&last_ofnbr,&ofnbr));
 if (i == 0  || i == -1 || i == -2 || i == -3)
  {
      /* creation d'une fiche client pour la premiere fois */
     pcli  = (S_CLIENT *)malloc(sizeof(S_CLIENT));
      init_fcli(0,pcli);
      title("FICHE DU CLIENT",10);
      existe = 0;
  }
 else if (i == 1)
```

```
    {       /* si une offre est deja cree */
     if ( ofnbr != 0)
        {
         sprintf(fname,"c%05d.dat",ofnbr);
         num = load_f(fname);
         if (num == -3)
          {
             /* le fichier contenant les donnees n'existe pas */
           sound_bell();
           print_warning(32,RED,0.0);
           getch();
           return;
          }
         if (num == 0 || num == 1 || num == 2  || num == 3 || num == 4)
          {
           if (pstruc[0].nbr_unites != 0)
              {
               /* a chaque lecture du fichier, toutes les structures */
               /* chargees, on libere toutes celles dont on a pas besoin*/
               free(pbu);
               free(pbs);
               free(pNV);
               if (pstruc[0].nbr_autos > 0) free(pfsc_con);
               if (pstruc[0].nbr_isobard > 0) free(pisobar);
              }
           free(pstruc);
           title("FICHE DU CLIENT",pcli[0].offernbr);
           existe = 1;
          }
         else return;
        }
    }
/* initialisation du menu */

 get_main_cli(&nm,&mmenu);



/** dessin du menu principal  et de la souris     **/
 init_drawbox(0,0);
 for (j=1 ; j < nm ; j++)
   draw_box(j,mmenu,cmen,cmtx,mmenu[j].text);

 hello_mickey();
 set_mouse_sensitivity(4,8);
 mouse_display();
```

```
/** initialisation des variables   **/
 menu_box  = -1;
field_box = -1;
bmouse    = 0;
action    = 0;
ancien    = -1;
maxx      = 0;
ginp      = 0;
maxy      = 0;

while(1)
 {
   kbgetkeymouse(&ps,&key,&bmouse,&xmouse,&ymouse);
   if (bmouse != 0)
     {
      if ((menu_box = box_number(xmouse,ymouse,mmenu,nm,0,0)) != -
1)
        {
         if (action == EDITION)
           {
            mouse_hide();
            sprintf(fname,"c%05d.dat",pcli[num_struct].offernbr);
            sauver_f(fname);
            /* sauvegarde des dernieres modifications et liberation */
            /* des structures */
            if (pstruc->nbr_unites > 0)
               free(pbu);
            if (pstruc->nbr_bs > 0)
               free(pbs);
            if (pstruc->nbr_ch > 0)
               free(pNV);
            if (pstruc->nbr_autos > 0)
               free(pfsc_con);
            if (pstruc->nbr_isobard > 0)
               free(pisobar);
          free(pstruc);
            ginp = 0;
            action = 0;
            mouse_display();
           }
         if (menu_box != ancien)
           {
            mouse_hide();
            init_drawbox(0,0);
```

```
draw_box(menu_box,mmenu,cmbox,cmtx,mmenu[menu_box].text);
                if (ancien != -1)
                  draw_box(ancien,mmenu,cmen,cmtx,mmenu[ancien].text);
                ancien = menu_box;
                init_drawbox(x,y);
                mouse_display();
               }
          }
        else if (action == EDITION && ginp == 1)
          {
           if ((field_box = box_number(xmouse,ymouse,bdata,nline,x,y)) != -
1)
               {
                mouse_hide();
                if (field_box == 1 || field_box == 2 || field_box == 4)
                 {
                  /* les cas "1","2","4", correspondent aux dates  et */
                  /* au numero d'offre, qui ne peuvent etre modifies. */
                  sound_bell();
                  message_err(1,7);
                  ginp = 1;
                  mouse_display();
                 }
                else
                 {
                  encode_value(string,field_box,num_struct);
                  draw_box(field_box,bdata,cwactbox,cmtx,string);
                  draw_message(field_box,bdata);
                  ginp = 2;
                 }
               }
          }
      }
    else if (!bmouse)
      {
       if (menu_box != -1)
         {
          if (strcmp(mmenu[ancien].fmt,"END") == 0)
            action = RETURN;
          else if (strcmp(mmenu[ancien].fmt,"EDITION") == 0)
            action = EDITION;
          else if (strcmp(mmenu[ancien].fmt,"CREATION") == 0)
            action = CREATION;
          else if (strcmp(mmenu[ancien].fmt,"SELECTION") == 0)
            action = SELECTION;
```

```
    else if (strcmp(mmenu[ancien].fmt,"COPIE") == 0)
      action = COPIE;
    else if (strcmp(mmenu[ancien].fmt,"DESSIN") == 0)
      action = DESSIN;
    menu_box = -1;
   }
  else if (action == CREATION)
   {
    mouse_hide();
    init_drawbox(x,y);
    get_window_cli(bdata,nline);
    window_size(x,y,nline,bdata,&maxx,&maxy);
    draw_menu(x,y,nline,bdata,maxx,maxy,0);
    mouse_display();
    if (read_data_cli(bdata,nline,field_box,num_struct,action) == -1)
      {
       setfillstyle(1,cscr);
       bar(30,60,610,440);
       mouse_display();
      }
    action = 0;
   }
  else if (action == EDITION && field_box == -1 && ginp == 0)
   {
    if (existe == 0)  /* si pas de strucure client active */
      {
       message_err(11,7);
       action = 0;
      }
    else if (existe == 1)
      {
       sprintf(fname,"c%05d.dat",pcli[num_struct].offernbr);
       free(pcli);
       i = load_f(fname);
       if (i != -1 && i != -2)
        {
         mouse_hide();
         get_window_cli(bdata,nline);
         window_size(x,y,nline,bdata,&maxx,&maxy);
         draw_menu(x,y,nline,bdata,maxx,maxy,0);
         ginp = 1;
         mouse_display();
        }
       else
        {
         /* si aucune fiche client n'est introduite */
```

```
            sound_bell();
            message_err(8,7);
            action = 0;
          }
       }
    }
  else if (action == EDITION && field_box != -1 && ginp == 2)
    {
     read_data_cli(bdata,nline,field_box,num_struct,action);
     field_box = -1;
     ginp = 1;
     mouse_display();
    }
  else if (action == SELECTION)
    {
     if (last_ofnbr > 10)
        {
         /* si plus d'une offre sont crees */
         n = get_selection(&ofnbr,&string);
         if (n == 1 || n == 0)
          {
            mouse_hide();
            sprintf(fname,"c%05d.dat",ofnbr);
            title("FICHE CLIENT",ofnbr);
            get_main_cli(&nm,&mmenu);
            init_drawbox(0,0);
            for (j=1 ; j < nm ; j++)
               draw_box(j,mmenu,cmen,cmtx,mmenu[j].text);
            mouse_display();
            if (load_f(fname));
               {
                init_drawbox(x,y);
                mouse_hide();
                get_window_cli(bdata,nline);
                window_size(x,y,nline,bdata,&maxx,&maxy);
                draw_menu(x,y,nline,bdata,maxx,maxy,0);
                mouse_display();
                if (pstruc->nbr_unites != 0) free(pbu);
                if (pstruc->nbr_bs != 0) free(pbs);
                if (pstruc->nbr_ch != 0) free(pNV);
                if (pstruc->nbr_autos != 0) free(pfsc_con);
                if (pstruc->nbr_isobard > 0) free(pisobar);
                free(pstruc);
                init_drawbox(0,0);
                action = 0;
               }
```

```
        }
       else
        {
         /* si aucune offre n'a ete selectionnee */
          message_err(3,7);
          action = 0;
        }
      }
 else
    {
     /* si aucune offre n'a ete cree */
     message_err(10,7);
     action = 0;
     mouse_display();
    }
}
else if (action == COPIE)
 {
 if (existe == 0)
    {
     /* si pas d'offre active */
     message_err(11,7);
     action = 0;
    }
  else if (existe == 1)
    {
     j = control_copy();
     if (j == 1)
      {
       /* rendre active l'offre copiee */
       title("FICHE DU CLIENT",pcli[0].offernbr);
       get_main_cli(&nm,&mmenu);
       init_drawbox(0,0);
       for  (j=1 ; j < nm ; j++)
       draw_box(j,mmenu,cmen,cmtx,mmenu[j].text);
       message_err(6,7);
      }
    else /* si copie annullee, afficher message */
       message_err(7,7);
     init_drawbox(x,y);
     get_window_cli(bdata,nline);
     window_size(x,y,nline,bdata,&maxx,&maxy);
     draw_menu(x,y,nline,bdata,maxx,maxy,0);
     init_drawbox(0,0);
   }
     action = 0;
```

```
            mouse_display();

          }
        else if (action == DESSIN)
          {
            sprintf(fname,"c%05d.dat",pcli[0].offernbr);
            free(pcli);
            i = load_f(fname);
            if (i != 0 && i != -1 && i != -2)
              {
                /* si l'offre est cree et que au moins les donnees */
                /* du batiment principal sont introduites          */
                mouse_hide();
                dr_project(pstruc[0].nbr_unites,pbu,pbs,&pfsc,pfsc_con,1);
                mouse_display();
                if (pstruc->nbr_unites > 0) free(pbu);
                if (pstruc->nbr_bs > 0) free(pbs);
                if (pstruc->nbr_ch > 0) free(pNV);
                if (pstruc->nbr_autos > 0) free(pfsc_con);
                if (pstruc->nbr_isobard > 0) free(pisobar);
              free(pstruc);
              }
            else
              {
                /* si les donnees sur batiment principal */
                /* ne sont pas introduites               */
                sound_bell();
                message_err(5,7);
              }
            action = 0;
          }

        else if (action == RETURN)
          {
            save_last("last_ofn.dat",last_ofnbr,pcli->offernbr);
            restorecrtmode();
            free(pcli);
            free(bdata);
            exit(0);
          }
      }
  }

}

/*****************************************************************************/
```

```
/*        Introduction et modification des donnees              */
/*************************************************************************/
int read_data_cli(ln,nline,field_box,num_struct,action)
S_MENU   *ln;
int nline,num_struct,action,field_box;

{
 char string[50];
 char fname[15];
 int j;
 if (action == 2)
    read_data_ch(field_box,ln,nline);
 else if (action == 1)
  {
    if (read_data_add(ln,nline,num_struct) == 1)
     {
        /* si une nouvelle offre est cree, cett offre devient active */
        last_ofnbr = pcli[num_struct].offernbr;
        existe = 1;
        mouse_hide();
        title("FICHE CLIENT",last_ofnbr);
        get_main_cli(&nm,&mmenu);
       init_drawbox(0,0);
        for (j=1 ; j < nm ; j++)
          draw_box(j,mmenu,cmen,cmtx,mmenu[j].text);
        sprintf(fname,"c%05d.dat",pcli[num_struct].offernbr);
        /* Quand une fiche client est cree, on initialise la structure */
        /* pstruc a 0 pour pouvoir controler l'existence ou non des    */
        /* structures S_BU,S_BS, et S_NV. Et ne faire de free de ces   */
        /* structures que si elles existent.                          */
        pstruc = (S_NBSTRUC  *)malloc(sizeof(S_NBSTRUC));
        pstruc[0].nbr_unites = 0;
        pstruc[0].nbr_bs = 0;
        pstruc[0].nbr_ch = 0;
        pstruc[0].nbr_autos = 0;
        pstruc[0].nbr_isobard = 0;
        sauver_f(fname);
        free(pstruc);
        /* sauver la nouvelle offre active et le dernier numero d'offre*/
        save_last("last_ofn.dat",last_ofnbr,pcli->offernbr);
        mouse_display();
     }
   }
}
/*************************************************************************/
/*   Creation d'une fiche client.                                        */
```

```
/*****************************************************************************/
int read_data_add(ln,nline,num_struct)
S_MENU   *ln;
int nline,num_struct;

{
 int num_err,ok,n,i;
 char string[50];
 char str[50];
 int it = 4;
 string[0] = '\0';
 str[0] = '\0';

 while (it < nline)
  {
    while (it <nline && ln[it].nmax <= 0) {it++;}
    if (it == nline) break;
    if (it == 4 && last_ofnbr != 0)
        pcli[0].offernbr = last_ofnbr + 10;
    if (it == 4)
      {
       encode_value(string,it,num_struct);
       draw_box(it,ln,cwbox,cwtx,string);
      }
    else
      {
       encode_value(string,it,num_struct);
       draw_box(it,ln,cwactbox,cwtx,string);
      }
    draw_message(it,ln);
    if (ln[it].menu == 0)
      {
       mouse_hide();
       n = get_value_cli(ln[it].posdx+x,ln[it].posdy+y,0,string,ln[it].nmax,it);
       if (n == -1) return(-1);
       if (n == -2)
        {
         /* si RETURN dans le cas des remarques */
         string[0] = '\0';
         for (i = 9;i <= 12;i++)
           {
            decode_value(string,i,num_struct);
            encode_value(string,i,num_struct);
            draw_box(it,ln,cwbox,cwtx,string);
           }
         it = 13;
```

```
          }
        if (string[0] == '\b')
         {
           strcpy(str,string);
           message_err(0,7);
           encode_value(string,it,num_struct);
           draw_box(it,ln,cwbox,cwtx,string);
           it = it == 4 ? it : it - 1;
           while (ln[it].nmax <= 0 && it > 1) it--;
          encode_value(string,it,num_struct);
           draw_box(it,ln,cwbox,cwactbox,string);
         }
        if (str[0] != '\b')
           decode_value(string,it,num_struct);
        ok = validate_cli(it,string,&num_err);
        if (ok)
         {
           message_err(0,0);
           encode_value(string,it,num_struct);
           draw_box(it,ln,cwbox,cwtx,string);
           it++;
         }
        else if (!ok)
         {
           sound_bell();
           message_err(num_err,7);
         }
        /*strcpy(str,'\b');*/
        str[0] = '\b';
        mouse_display();
        }
       else if (ln[it].menu == 1)
        {
         if (pop_up(&it,0,x,y,pcli) == -1) return(-1);
         it++;
        }
    }
 mouse_display();
}

/*************************************************************************/
/*       modification de donnees                                         */
/*************************************************************************/
int read_data_ch(it,ln,nline)
int it,nline;
S_MENU  *ln;
```

```
{
 int num_err,ok;
 char string[50];

 if (ln[it].menu == 1)
   pop_up(&it,0,x,y,pcli);
 else if (ln[it].menu == 0)
       {
        while (it)
            {
             mouse_hide();
             encode_value(string,it,0);
             draw_box(it,ln,cwactbox,cwtx,string);
             draw_message(it,ln);

get_value_cli(ln[it].posdx+x,ln[it].posdy+y,0,string,ln[it].nmax,2);
             draw_box(it,ln,cwactbox,cwtx,string);
             decode_value(string,it,0);
             ok = validate_cli(string,&num_err);
             if (ok)
              {
                message_err(0,0);
                encode_value(string,it,0);
                draw_box(it,ln,cwbox,cwtx,string);
                it = 0;
              }
             else
                {
                   message_err(num_err,7);
                }
            }
       }
 mouse_display();
}

/*************************************************************************/
/***            encodage de donnees                    *****/
/*************************************************************************/
int encode_value(string,box_num,i)
char *string;
int box_num,i;
{
 char str[10];
 struct date today;
 str[0] = '\0';
```

```
switch(box_num)
 {
  case 1 : /* date du jours = date du systeme */
   getdate(&today);

sprintf(str,"%02d:%02d:%4d",today.da_day,today.da_mon,today.da_year);
   strcpy(string,str);
   break;

  case 2 : /* date de creation */
   strcpy(string,pcli[i].date);
   break;

  case 4 :
   sprintf(string,bdata[box_num].fmt,pcli[i].offernbr);
   break;

  case 5 :
     strcpy(string,pcli[i].cliname);
     break;
  case 6 :
   strcpy(string,pcli[i].prjname);
   break;

  case 7 :
   strcpy(string,pcli[i].branch);
   break;

  case 8 :
   strcpy(string,pcli[i].blduse);
   break;

  case 9 :
   strcpy(string,pcli[i].remarques[0]);
   break;

  case 10 :
   strcpy(string,pcli[i].remarques[1]);
   break;

  case 11 :
   strcpy(string,pcli[i].remarques[2]);
   break;

  case 12 :
```

```
      strcpy(string,pcli[i].remarques[3]);
      break;

    case 13 :
      strcpy(string,pcli[i].remarques[4]);
      break;
    default :
      break;

   }
}

/*****************************************************************************/
/***          decodage des donnees                                         ***/
/*****************************************************************************/
int decode_value(string,box_num,i)
char  *string;
int i,box_num;

{
 float bid;
 int j;
 char fmt[] = "% ";
 fmt[1] = bdata[box_num].fmt[strlen(bdata[box_num].fmt)-1];
 sscanf("111.2","%f",&bid);

 switch(box_num)
  {
    case 1 : /* la date du jours est seulement une information */
             /* pour l'utilisateur pendant son travail       */
      break;

    case 2 :
      strcpy(pcli[i].date,string);
      break;

    case 4 :
      sscanf(string,fmt,&(pcli[i].offernbr));
      break;

    case 5 :
       strcpy(pcli[i].cliname,string);
       break;

     case 6 :
       strcpy(pcli[i].prjname,string);
```

```
        break;

      case 7 :
        strcpy(pcli[i].branch,string);
        break;

      case 8 :
        strcpy(pcli[i].bIduse,string);
        break;

     case 9 :
        strcpy(pcli[i].remarques[0],string);
        break;

     case 10 :
        strcpy(pcli[i].remarques[1],string);
        break;

     case 11 :
        strcpy(pcli[i].remarques[2],string);
        break;

     case 12 :
        strcpy(pcli[i].remarques[3],string);
        break;

     case 13 :
        strcpy(pcli[i].remarques[4],string);
        break;
     default :
         break;

    }
}

/*************************************************************************/
/*       initialisation de la structure                                  */
/*************************************************************************/
int init_fcli(i,pcli)
int i;
S_CLIENT  *pcli;

{
  char str[12];
  struct date today;
  int j;
```

```
 getdate(&today);
 sprintf(str,"%02d:%02d:%4d",today.da_day,today.da_mon,today.da_year);
 strcpy(pcli[i].date,str);

pcli[i].offernbr = last_ofnbr + 10;

 pcli[i].cliname[0] =  '\0';

 pcli[i].prjname[0] = '\0';

 pcli[i].branch[0]  = '\0';

 pcli[i].bldUse[0]  = '\0';

 for (j=0 ; j < 5 ; j++)
       strcpy(pcli[i].remarques[j]," ");
}

/*****************************************************************************/
/*              les pop_up menus                                             */
/*****************************************************************************/
int pop_up(it,i,x,y,pcli)
int i,*it,x,y;
S_CLIENT  *pcli;

#define BXO bdata[*it].bx + bdata[*it].dx + x
#define BYO bdata[*it].by + y

{
 char *choice;
 char string[50];
 int num_err,ok,erreur;

 switch(*it)
  {
    case 7 :
     erreur = get_choice(BXO,BYO,bldusetyp,&choice,0,1);
     break;

    case 8 :
       erreur = get_choice(BXO,BYO,branchlist,&choice,0,1);
       break;

    default :
        return;
```

```
  }

if (erreur == -1) return(-1);
decode_value(choice,*it,0);
encode_value(string,*it,0);
init_drawbox(x,y);
draw_box(*it,bdata,cwbox,cwtx,string);
ok = validate_cli(*it,choice,&num_err);
if (ok != 0)  message_err(num_err,7);
return(1);
}

/*************************************************************************/
/*                Selection d'une offre                                  */
/* Parcours de tous les fichiers existants pour rechercher les donnees  */
/* necessaires a la creation d'une liste des offres crees. La liste     */
/* est partitionnee en des sous listes des 10 offres maximum. Les listes*/
/* sont affichees une par une.                                          */
/*************************************************************************/
int get_selection(lastnbr,choice)
int  *lastnbr;
char *choice;
#define BXO 50 + x
#define BYO 50 + y
{

 float bid;
 char string[25];
 char *str;
 int dim,i,k,test,j;
 char fname[15];
 S_OFLIST  *poflist; /* chaque structure contiendra les informations   */
                     /* necessaires qui seront affichees. L'ensemble   */
                     /* de ces structures forment la liste des offres. */
 char *pointer[13];
 char fmt[] ="% ";
 fmt[1] = bdata[2].fmt[strlen(bdata[2].fmt)-1];
 sscanf("111.2","%f",&bid);

 j = 10;
 test = j;
 /* Test pour voir s'il y a plus de 10 offres crees. */
 if (((last_ofnbr) / 10) > 10)  dim = (int) min(((last_ofnbr) / 10),10);
 else dim = (last_ofnbr) / 10;
 while (dim <= (last_ofnbr / 10))
```

```
{
 poflist = (S_OFLIST *)malloc((dim+2)*sizeof(S_OFLIST));
 mouse_hide();
 /* affichage du titre */
 window_selection(bdata,3);
 window_size(x,y,3,bdata,&maxx,&maxy);
 draw_fenetre(x,y,maxx,maxy);
 setcolor(cwatx);
 outtextxy(160,110,bdata[1].text);
 outtextxy(180,120,"_____");
 if ( j > 10*10)
     strcpy(poflist[0].projet,"       page precedente");
 else
     strcpy(poflist[0].projet,"    **************************");
 pointer[0] = &(poflist[0].projet);
 k = 1;
 mouse_display();
 while (test/10 <= dim) /* boucle pour parcourir tous les fichiers */
                        /* existents                              */
   {
    sprintf(fname,"c%05d.dat",j);
    i = load_f(fname);
    if (i == 1 || i == 2 || i == 0 || i == 3)
     {
      strcpy(string,pcli[0].prjname);
      string[15] = '\0';
      /* construction de la liste des offres */
      if (i==1 || i == 2 || i == 3)
      sprintf(poflist[k].projet,"%6d     %s     %15s     %s
",j,pcli[0].date,string,pbu[0].bldtyp);
      else if (i == 0)
       sprintf(poflist[k].projet,"%6d     %s     %15s     %s
",j,pcli[0].date,string,(NULL));
      pointer[k] = &(poflist[k].projet);
      k++;
      if (pstruc->nbr_unites != 0) free(pbu);
      if (pstruc->nbr_bs != 0)     free(pbs);
      if (pstruc->nbr_ch != 0)     free(pNV);
      if (pstruc->nbr_autos != 0)  free(pfsc_con);
      if (pstruc->nbr_isobard > 0) free(pisobar);
      free(pstruc);
      free(pcli);
     }
    else
     {
      strcpy(poflist[k].projet," ");
```

```
            pointer[k] = &(poflist[k].projet);
          }
        j += 10;
        test += 10;
  }
if ( (last_ofnbr - j) / 10 >= 0)
    strcpy(poflist[k].projet,"       page suivante");
else
    strcpy(poflist[k].projet,"    **************************");
pointer[k] = &(poflist[k].projet);
k++;
while (k < 12)
  {
    strcpy(poflist[k].projet, " ");
    pointer[k] = &(poflist[k].projet);
    k++;
  }
mouse_hide();
get_choice(BXO,BYO,pointer,&choice,dim+2,3);
mouse_display();
    if (strcmp(choice,"       page suivante") == 0)
  {
    test = 10;
    dim = (int) min(10,(last_ofnbr - j)/10 + 1);
    k = 0;
    free(poflist);
  }
else if (strcmp(choice,"       page precedente") == 0)
  {
    j = j - (dim + 10)*10 ;
    test = 10;
    dim = 10;
    k = 0;
    free(poflist);
  }
else if (strcmp(choice,"    **************************") == 0
                || strcmp(choice,"    **************************") == 0)
        return(0);
else
  {
    sscanf(choice,"%d",lastnbr);
    free(poflist);
    mouse_display();
    dim = 100; /* pour sortir de la boucle */
    return(1);
  }
```

```
  }
 }

/*****************************************************************************/
int message_err(num_err,color)
int num_err;
int color;

{
 int bkcolor,frcolor;
 bkcolor = getpixel(1,430);
 frcolor = getcolor();
 setfillstyle(1,bkcolor);
 bar(1,430,638,450);
 if (!color) return;
 setcolor(color);
 switch(num_err)
  {
   case 1 :
       outtextxy(110,435,"Cette donnee ne peut etre modifiee");
       break;
   case 2 :
       outtextxy(110,435,"Cette valeur est obligatoire");
       break;
   case 3 :
       outtextxy(110,435,"L'offre active est la seule qui existe");
       break;

   case 4 :
       outtextxy(110,435,"Donnees relatives a l'offre selectionnee ");
       break;

   case 5 :
       outtextxy(110,435,"Introduire d'abord les donnees sur le batiment");
       break;

   case 6 :
       outtextxy(150,435," La copie est faite");
       break;

   case 7:
       outtextxy(150,435,"La copie est annulee");
       break;

    case 8 :
        outtextxy(110,435,"Introduire d'abord la fiche du client SVP");
```

```
        break;

    case 9 :
       outtextxy(110,200,"VEUILLEZ VOUS ASSUREZ QUE LE FICHIER
EXISTE.");
       outtextxy(125,220,"APPUYEZ SUR UNE TOUCHE POUR CONTINUER
SVP:");
        break;

    case 10 :
      outtextxy(110,435,"aucune offre n'a ete selectionnee");
      break;

    case 11:
      outtextxy(110,435,"introduire d'abord une offre");
      break;

   default :
        outtextxy(110,435,"");

  }
 setcolor(frcolor);
}

/*****************************/
int validate_cli(it,string,num)
char *string;
int it;
int *num;
{
 if (string[0] == '\0'  && it < 9 )
  {
   *num = 2;
   return(0);
  }
 else return(1);
}

/*****************************************************************************
/
/*     confirmation avant de copier une offre dans une nouvelle          */
/*****************************************************************************/
control_copy()


{
```

```
 int bmouse,ps,key,k,bld_unit,xmouse,ymouse;
 int x0,y0;
 x0 = x + 160;
 y0 = y + 70;
 draw_fenetre(x0,y0,x0+220,y0+140);
 setcolor(BLACK);
 setfillstyle(1,LIGHTGRAY);
 outtextxy(x0+70,y0+22,"Confirmez");
 bar3d(x0+33,y0+60,x0+103,y0+105,0,0);
 outtextxy(x0+50,y0+80,"OUI");
 bar3d(x0+115,y0+60,x0+185,y0+105,0,0);
 outtextxy(x0+132,y0+80,"NON");
 setcolor(WHITE);
 moveto(x0+103,y0+60); lineto(x0+33,y0+60); lineto(x0+33,y0+105);
 moveto(x0+185,y0+60); lineto(x0+115,y0+60), lineto(x0+115,y0+105);
 mouse_display();
 bmouse = 0;
 k = 0;
 while(!k)
 {
  kbgetkeymouse(&ps,&key,&bmouse,&xmouse,&ymouse);
  if (bmouse)
    {
     if (xmouse > x0+115 && xmouse < x0+185 &&
         ymouse > y0+60 && ymouse < y0+105) k = 2;
     else if(xmouse > x0+33 && xmouse < x0+103 &&
             ymouse > y0+60 && ymouse < y0+105) k = 1;
     else k = 0;
    }
 }
 mouse_hide();
 if (k == 1)
    copy_offre();
 else return(0);
}

/*****************************************************************************/
copy_offre()

{
 int i;
 char fname[15];
 char str[50];
 struct date today;
 sprintf(fname, c%05d.dat ,pcli[0].offernbr);
 i = load_f(fname);
```

```
if (i != -2 && i != -3 && i != -4)
 {
   pcli[0].offernbr = last_ofnbr + 10;
   getdate(&today);
   sprintf(str,"%02d:%02d:%4d",today.da_day,today.da_mon,today.da_year);
   strcpy(pcli[0].date,str);
   last_ofnbr += 10;
   sprintf(fname,"c%05d.dat",pcli[0].offernbr);
   sauver_f(fname);
   if (pstruc[0].nbr_unites > 0) free(pbu);
   if (pstruc[0].nbr_bs > 0) free(pbs);
   if (pstruc[0].nbr_ch > 0) free(pNV);
   if (pstruc[0].nbr_autos > 0) free(pfsc_con);
   if (pstruc[0].nbr_isobard > 0) free(pisobar);
   free(pstruc);
 }
 return(1);
}
/*****************************************************************************/
```

```
/*****************************************************************************/
/* Fonction principal d'introduction, modification ou suppression d'une  */
/* unite de bardage interieur, d'isolation ou de sous basement.        */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include <dos.h>

#include "\cyprion\color.h"
#include "\cyprion\bldunit.h"
#include "\cyprion\valid.h"
#include "\cyprion\bayspa.h"
#include "\cyprion\mouse.h"
#include "\cyprion\datamenu.h"
#include "\cyprion\wu\charges.h"
#include "\cyprion\coordfsc.h"
#include "\cyprion\fsc_cond.h"
#include "\cyprion\nb_struc.h"
#include "\cyprion\isol\limite.h"
#include "\cyprion\wu\print_wa.h"
#include "\cyprion\cli.h"
#include "\cyprion\iso_bar.h"

#define SELECTION  1  /** selection d'un batiment et d'un cote  **/
#define ADD    2
#define CHANGE 3
#define DELETE  4
#define RETURN 6
#define ZOOM   7
#define DESSIN 5
#define extern unsigned _stklen = 10000

int get_value_pad();
int mycolors();
int read_data_add();
int read_data_ch();
int get_val_string();
int get_choice();
int chercher();
```

```
S_AXE *paxe;

S_BU    *pbu;
S_CLIENT *pcli;
S_NV    *pNV;
S_BS    *pbs;
S_NBSTRUC   *pstruc;
S_MENU    *pmenu;
S_MENU    *mmenu;
S_MENU    *bdata;
S_MENU    *ln;
S_FSC          *pfsc;
S_FSC_CON  *pfsc_con;
S_CLIENT   *pcli;
S_ISOL_BARD  *pisobar;


int xmouse,ymouse,bmouse,action;
int num_ref,k;
int  max_unit = 1,bld_unit,isol_unit;
int  nm;
int op = 0;
int nline;
int x = 100, y = 100;
int dx0,dx1,dy1,dy0;
float ds1,ds2,ds3;
int type;  /* type de traitement :isolation, bardage interieur ou sous */
        /* basement.                                          */
int nbr;   /* nombre d'axes du cote actif.                       */
int posx,maxx,maxy,posy,dx,dy,num_err,menu,nmax;
int fr;    /* pour tester si les la structures des coordonnees des axes */
        /* est cree ou non. */


main(argc,argv)
int argc;
char *argv[];
{
 int gdrive = DETECT,gmode;
 int key_pad;
 int i,out,nbr_unites;
 int select,n;
 char side,ch;
 char string[50];
 char fname[15];
 float limite;
```

```
int j,introduit ;
int x0,y0,x1,y1;
int color ;
int ancien,ginp,last,field_box,menu_box,key,ps;

initgraph(&gdrive,&gmode,"");

if (strcmp(argv[2],"1") == 0)      type = 1; /* isolation        */
else if (strcmp(argv[2],"2") == 0) type = 2; /* bardage interieur */
else if (strcmp(argv[2],"3") == 0) type = 3; /* sous basement     */
if (type == 1) nline = 9;
else if (type == 2) nline = 7;
else if (type == 3) nline = 4;
cleardevice();
mycolors();
draw_frame(0,0,639,479,BLACK);

paxe = (S_AXE *)malloc(0*sizeof(S_AXE));
fr = 0;
bdata = (S_MENU *)malloc(nline*sizeof(S_MENU));
num_ref = 10;

i = load_last("last_ofn.dat",&j,&num_ref);
if (i != 1)
 {
   print_warning(32,RED,0.0);
   free(bdata);
   free(pstruc);
   free(pNV);
   out = getch();
   if (!out) getch();
   return;
 }
else
   sprintf(fname,"c%05d.dat",num_ref);
i = load_f(fname);
if (i != 2  && i != 3 && i != 4)   /* Cas ou le fichier n'existe pas */
 {
   print_warning(32,RED,0.0);
   free(bdata);
   free(pstruc);
   free(pNV);
   out = getch();
   if (!out) getch();
   return;
 }
```

```
if (i != 2  && i != 0 )
 {
   max_unit = 0;
 }
else
 {
   max_unit = pstruc->nbr_isobard ;
 }
if (type == 1)
  title("Isolation",num_ref);
else if (type == 2)
  title("Bardage Interieur",num_ref);
else if (type == 3)
  title("Sous Basements",num_ref);

nbr_unites = pstruc->nbr_unites;

/***initialisation du menu principal ***/
get_main_isobar(&nm,&mmenu);

/***dessin du menu principal***/
init_drawbox(0,0);
for (j = 1 ; j < nm ;j++)
  draw_box(j,mmenu,cmen,cmtx,mmenu[j].text);

/*** initialisation de la souris***/
hello_mickey();
set_mouse_sensitivity(4,8);
mouse_display();

mouse_hide();
dr_project(nbr_unites,pbu,pbs,&pfsc,pfsc_con,1);
mouse_display();
/*** initalisation des variables***/
menu_box  = -1;
isol_unit = -1;
bld_unit  = -1;
field_box = -1;
bmouse    = 0;
action    = 0;
ancien    = -1;
ginp = 0;
maxx = 0;
maxy = 0;

/*****************************************************************************/
```

```
/* Principe de gestion des differentes actions:                    */
/*   - quelque soit l'action a realiser, d'abord selectionner un cote */
/*     d'un batiment. Ceci avec l'option "SELECTION" du menu. Ce cote */
/*     reste actif tant qu'une autre selection n'est pas faite.      */
/*   - apres des modifications, l'option "DESSIN" du menu permet de  */
/*     visualiser le cote modifie.                                   */
/*****************************************************************/
while(1)
 {
  kbgetkeymouse(&ps,&key,&bmouse,&xmouse,&ymouse);
  if (bmouse != 0)
    {
     if ((menu_box = box_number(xmouse,ymouse,mmenu,nm,0,0)) != -1)
       {
          free_memory();
          if (bld_unit != -1)
           {
            if (ancien == 5 && menu_box == 2)
                 ginp = 2;
            else if (ancien == 5 && (menu_box == 3 || menu_box == 4))
                ginp = 2;
            if (ancien == 4 && menu_box == 4)
                 ginp = 2;
            if (ancien == 3 && ( menu_box == 3 || menu_box == 4))
                 ginp = 2;
            else if (ancien == 3 && menu_box == 2)
                 ginp = 2;
            if (ancien == 4 && menu_box == 2)
              ginp = 2;
            if (ancien == 4 && menu_box == 3)
             ginp = 2;
             if ((ancien == 2 || ancien == 3 || ancien == 5 || ancien == 1)
                          && menu_box == 1)
               ginp = 0;
           }
          if (menu_box != ancien)
           {
            mouse_hide();
            init_drawbox(0,0);
           draw_box(menu_box,mmenu,cmbox,cmtx,mmenu[menu_box].text);
            if (ancien != -1)
                draw_box(ancien,mmenu,cmen,cmtx,mmenu[ancien].text);

            ancien = menu_box;
            init_drawbox(x,y);
            mouse_display();
```

```
    }
}
  else if (action == SELECTION && ginp == 1 )
    {
     bld_unit = get_unit(xmouse,ymouse,pbu,nbr_unites,&side,2);
     if (bld_unit != -1)
      {
        if (ch == 'T')
            side = 'T';

        /* seul le batiment principal ou les appentis peuvent */
        /* etre traites.                                      */
        if (pbu[bld_unit].bldtyp != "RE" && pbu[bld_unit].bldtyp != "PA"
                  && pbu[bld_unit].bldtyp != "CP")
         {
          print_warning(0,WHITE,0.);
          mouse_hide();
         /**** nbr = nombre des axes ***/
          if (side == 'L' || side == 'R' || side == 'T' || side == 'X')
           {
            if (bld_unit == 0)
               nbr = pbu[bld_unit].nbrtrv + 1;
            else if (bld_unit > 0)
               nbr = pbu[bld_unit].tobay - pbu[bld_unit].frbay + 2;
           }
          else if (side == 'F')
              nbr = pbu[bld_unit].nbayf + 1;
          else if (side == 'B')
              nbr = pbu[bld_unit].nbayb+1;
          /* faire un free de paxe avant d'allouer pour n'avoir */
          /* qu'une structure a la fois et qui  correspond au   */
          /* actif.                                             */
          if (fr == 1) free(paxe);
          fr = 0;
          paxe = (S_AXE *)malloc(nbr*sizeof(S_AXE));

          draw_isol(side,0,bld_unit,pisobar,pbu,0,LIGHTGRAY,nbr);
          for ( j = 0 ; j <  max_unit ; j++)
           {
             /* ne dessiner que les unites se trouvant sur le */
             /* cote actif du batiment selectionne.           */
             if (pisobar[j].side == side &&
                pisobar[j].bldunit_nbr == bld_unit &&
                pisobar[j].type == type)
               draw_isol(side,j,bld_unit,pisobar,pbu,1,YELLOW,nbr);
           }
```

```
                ginp = 2;
                action = 0;
                mouse_display();
               }
             else
               {
                message_err(10,7,0.);
                ginp = 0;
               }
            }
          }
        else if (action == CHANGE && ginp == 4)
         {
          if ((field_box = box_number(xmouse,ymouse,bdata,nline,x,y)) != -
1)
            {
              mouse_hide();
              encode_value(string,field_box,isol_unit);
              draw_box(field_box,bdata,cwactbox,cmtx,string);
              draw_message(field_box,bdata);
              ginp = 5;
            }
          }
        else if (action == ZOOM && !ginp)
         {
          ginp = 1;
          x0 = xmouse - 1;
          y0 = ymouse - 1;
         }
        else if (action == ZOOM && ginp == 2)
         {
          ginp = 3;
          x1 = xmouse - 1;
          y1 = ymouse - 1;
         }
        }

  else if (!bmouse)
   {
    if (menu_box !=-1)
      {
       if (strcmp(mmenu[ancien].fmt,"SELECTION") == 0)
            action = SELECTION;
       else if (strcmp(mmenu[ancien].fmt,"ADD") == 0)
            action = ADD;
       else if (strcmp(mmenu[ancien].fmt,"CHANGE") == 0)
```

```
        action = CHANGE;
    else if (strcmp(mmenu[ancien].fmt,"END") == 0)
        action = RETURN;
    else if (strcmp(mmenu[ancien].fmt,"ZOOM") == 0)
        action = ZOOM;
    else if (strcmp(mmenu[ancien].fmt,"DELETE") == 0)
        action = DELETE;
    else if (strcmp(mmenu[ancien].fmt,"DESSIN") == 0)
       action = DESSIN;
    if (action == ADD || action == CHANGE || action == DELETE
        || action == ZOOM || action == RETURN || action == DESSIN
        || action == SELECTION)
              menu_box  = -1;
  }
else if ((action == ADD || action == CHANGE || action == DELETE )
        && ginp == 0 )
  {
    sound_bell();
    message_err(9,7,0.);
    action = 0;
  }
else if (action == SELECTION && ginp == 0)
  {
    mouse_hide();
    if (type == 1)
      n = pop_up(20,0,0,isol_unit);
    else if (type == 2 || type == 3)
      n = 0;
    mouse_display();
    mouse_hide();
    dr_project(nbr_unites,pbu,pbs,&pfsc,pfsc_con,1);
    if (n == 0)
      {
        message_err(7,7,0.);
        ch = 'X';
        ginp = 1;
      }
    else if (n == 1)
      {
        message_err(13,7,0.);
        ch = 'T';
        ginp = 1;
      }
    else if (n == -1)
      {
        action = 0;
```

```
            ginp = 0;
          }
          mouse_display();
        }
      else if (action == ADD && ginp == 2 &&  bld_unit != -1 )
        {
          mouse_hide();
          if ( max_unit == 0)
           {
             pisobar = (S_ISOL_BARD *)malloc(sizeof(S_ISOL_BARD));
             isol_unit = 0;
             init_isol(isol_unit,pisobar);
           }
          else if (max_unit > 0)
           {
             pisobar = (S_ISOL_BARD *)realloc(pisobar,(max_unit +
1)*sizeof(S_ISOL_BARD));
             isol_unit = max_unit;
             init_isol(isol_unit,pisobar);
           }
          pisobar[isol_unit].side = side;
          pisobar[isol_unit].bldunit_nbr = bld_unit;
          pisobar[isol_unit].type = type;
          if (side == 'T')
           get_window_toiture(bdata,nline);
          else
           get_window_isol(bdata,nline);
          window_size(x,y,nline,bdata,&maxx,&maxy);
          draw_menu(x,y,nline,bdata,maxx,maxy,isol_unit);
          mouse_display();
          pisobar[isol_unit].bldunit_nbr = bld_unit;
          if (read_data_add(x,y,nline,bdata,pisobar,isol_unit,side) != -1)
           {
             int j = 2;
             maj_add(&j,isol_unit,side);
             mouse_hide();
             max_unit++;

      draw_isol(side,isol_unit,bld_unit,pisobar,pbu,0,LIGHTGRAY,nbr);
             for ( j = 0 ; j < max_unit - 1 ; j++)
              {
               if (pisobar[j].side == side &&
                  pisobar[j].bldunit_nbr == bld_unit &&
                  pisobar[j].type == type)
                draw_isol(side,j,bld_unit,pisobar,pbu,1,YELLOW,nbr);
              }
```

```
        draw_isol(side,isol_unit,bld_unit,pisobar,pbu,1,GREEN,nbr);
      pstruc[0].nbr_isobard = max_unit;
        sprintf(fname,"c%05d.dat",num_ref);
        sauver_f(fname);
        mouse_display();
      }
     else
      {
       setfillstyle(1,cscr);
       bar(30,60,610,440);
       action = 0;
       ginp = 0;
       bld_unit = -1;
       mouse_display();
      }
     action = 0;
    }

  else if ((action == CHANGE || action == DELETE) &&
           (ginp == 2 && bld_unit != -1))
   {
    message_err(11,7,0.);
    if (max_unit > 0)
     {
       get_isol(&isol_unit,max_unit,side,bld_unit);
       if (isol_unit != -1)
        ginp = 3;
       else  message_err(12,7,0.);
     }
    else
     {
       sound_bell();
       message_err(12,7,0.);
       action = 0;
     }
   }
  else if (action == CHANGE && field_box == -1 && ginp == 3 )
   {
    mouse_hide();
    if (side == 'T')
     get_window_toiture(bdata,nline);
    else
     get_window_isol(bdata,nline);
    window_size(x,y,nline,bdata,&maxx,&maxy);
    draw_menu(x,y,nline,bdata,maxx,maxy,isol_unit);
    mouse_display();
```

```
        ginp = 4;
       }
      else if (action == CHANGE && field_box != -1  && ginp == 5)
       {
        read_data_ch(field_box,x,y,bdata,isol_unit);
        maj_add(&field_box,isol_unit,side);
       pstruc[0].nbr_isobard = max_unit;
        sprintf(fname,"c%05d.dat",num_ref);
        sauver_f(fname);
        field_box = -1;
        ginp = 4;
       }
      else if (action == ZOOM && ginp == 1)
       {
        putpixel(x0,y0,WHITE);
        ginp = 1;
       }
      else if (action == ZOOM && ginp == 3)
       {
        setcolor(GREEN);
        mouse_hide();
        line(x0,y0,x1,y1);
        line(x1,y0,x1,y1);
        line(x1,y1,x0,y0);
        line(x0,y1,x0,y0);
       }
      else if (action == DELETE && ginp == 3 && isol_unit != -1)
       {
        message_err(11,7,0.);
        n = del_isobard(pbu,max_unit,pisobar,isol_unit);
        if (n == -1) return;
        if (n == 1)
         {
           mouse_hide();

      draw_isol(pisobar[isol_unit].side,isol_unit,pisobar[isol_unit].bldunit_n
br,pisobar,pbu,1,LIGHTGRAY,nbr);
           mouse_display();
           for (j = isol_unit ; j < max_unit - 1; j++)
            pisobar[j] = pisobar[j+1];
           max_unit--;
           pisobar = (S_ISOL_BARD
*)realloc(pisobar,(max_unit)*sizeof(S_ISOL_BARD));
         pstruc[0].nbr_isobard = max_unit;
           sprintf(fname,"c%05d.dat",num_ref);
           sauver_f(fname);
```

```
            message_err(0,0,0.);
          }
         else if (n == 0)
          {
            mouse_hide();

      draw_isol(pisobar[isol_unit].side,isol_unit,pisobar[isol_unit].bldunit_n
br,pisobar,pbu,1,YELLOW,nbr);
            mouse_display();
          }
         mouse_hide();
        draw_isol(side,0,bld_unit,pisobar,pbu,0,LIGHTGRAY,nbr);
        for ( j = 0 ; j <  max_unit ; j++)
          {
           if (pisobar[j].side == side &&
                 pisobar[j].bldunit_nbr == bld_unit &&
                              pisobar[j].type == type)
                draw_isol(side,j,bld_unit,pisobar,pbu,1,YELLOW,nbr);
          }
         mouse_display();
         action = 0;
        }
       else if (action == DESSIN)
        {
         if (bld_unit != -1 )
          {
            draw_isol(side,0,bld_unit,pisobar,pbu,0,LIGHTGRAY,nbr);
            for ( j = 0 ; j <  max_unit ; j++)
             {
              if (pisobar[j].side == side &&
                        pisobar[j].bldunit_nbr == bld_unit &&
                        pisobar[j].type == type)
                   draw_isol(side,j,bld_unit,pisobar,pbu,1,YELLOW,nbr);
             }
          }
         else
          {
            sound_bell();
            message_err(9,7,0.);
          }
         action = 0;
        }

       else if (action == RETURN)
        {
         /***sauvegarde des donnees ***/
```

```
            sprintf(fname,"c%05d.dat",num_ref);
            pstruc[0].nbr_isobard = max_unit;
            sauver_f(fname);
            free(pcli);
            free(pbu);
            free(pNV);
            free(pbs);
            free(paxe);
            free(bdata);
            if (pstruc->nbr_autos > 0) free(pfsc_con);
            if (pstruc->nbr_isobard > 0)
              free(pisobar);
            free(pstruc);
            restorecrtmode();
            exit(0);
          }
      }
    }
}

/*****************************************************************************/
/*              introduction des donnees                                     */
/*****************************************************************************/
read_data_add(x,y,nline,ln,pisobar,isol_unit,side)
int x,y,nline,isol_unit;
S_MENU      *ln;
S_ISOL_BARD *pisobar;
char side ;
{
 char string[50];
 char str[50];
 int ok,it = 1,n;
 float limite;
 string[0] = '\0';
 str[0] = '\0';

 while (it < nline)
  {
    while(it < nline && ln[it].nmax == 0) (it++;)
    if (it == nline) break;
    encode_value(string,it,isol_unit);
    draw_box(it,ln,cwactbox,cwtx,string);
    draw_message(it,ln);
    if (ln[it].menu == 0)
       {
         mouse_hide();
```

```
        n =
get_value_pad(maxx+10,y+10,ln[it].posdx+x,ln[it].posdy+y,0,string,ln[it].nma
x,1);
        if (n == -1) return (-1);
        if (n == 0 && string[0] != '\b') strcpy(string,"0");
        if (string[0] == '\b')
         {
          strcpy(str,string);
          message_err(0,7,0.);
          encode_value(string,it,isol_unit);
          draw_box(it,ln,cwbox,cwtx,string);
          it = it == 1 ? it : it - 1;
          while (ln[it].nmax <= 0 && it > 1) it--;
          encode_value(string,it,isol_unit);
          draw_box(it,ln,cwactbox,cwtx,string);
         }
        if (string[0] != '\0')
           decode_value(string,it,isol_unit);
        if (str[0] != '\b')
         {
          ok = validate_isol(it,isol_unit,&num_err,&limite,2);
          if (ok)
            {
             message_err(0,0,0.);

             encode_value(string,it,isol_unit);
             draw_box(it,ln,cwbox,cwtx,string);
             it++;
            }
           else
            {
             sound_bell();
             message_err(num_err,7,limite);
            }
         }
        str[0] = '\0';
        mouse_display();
       }
    else if (ln[it].menu == 1)
       {
        if (pop_up(it,x,y,isol_unit) == -1) return(-1);
        it++;
     }
   }
  mouse_display();
}
```

```
/*****************************************************************************/
/*      changement des donnees                                               */
/*****************************************************************************/
read_data_ch(it,x,y,ln,isol_unit)
int x,y,it,isol_unit;
S_MENU  *ln;

{
 int ok,n;
 float limite;
 char string[50];
 string[0] = '\0';
 if (ln[it].menu == 1)
  {
   pop_up(it,x,y,isol_unit);
  }
 else if (ln[it].menu == 0)
  {
   while (it)
     {
      encode_value(string,it,isol_unit);
      draw_box(it,ln,cwactbox,cwtx,string);
      draw_message(it,ln);
      n = get_value_pad(maxx+10,y+10,ln[it].posdx + x,ln[it].posdy +
y,0,string,ln[it].nmax,1);
      if (n == -1) return(-1);
      mouse_hide();
      draw_box(it,ln,cwactbox,cwtx,string);
      mouse_display();
      decode_value(string,it,isol_unit);
      ok = validate_isol(it,isol_unit,&num_err,&limite,3);
      if (ok)
       {
        mouse_hide();
        message_err(0,0,0.);
        encode_value(string,it,isol_unit);
        draw_box(it,ln,cwbox,cwtx,string);
        mouse_display();
        it = 0;
       }
      else
       {
        sound_bell();
        message_err(num_err,7,limite);
       }
```

```
            mouse_display();
          }
      }
    mouse_display();
}
/*****************************************************************************/
int init_isol(i,pisobar)
int i;
S_ISOL_BARD *pisobar;
{
  pisobar[i].type        = type;
  pisobar[i].bldunit_nbr  = pbu[i].bu_nb;
  pisobar[i].side        = ' ';
  pisobar[i].lenght       = 0;
  pisobar[i].level        = 0;
  pisobar[i].distance     = 0;
  pisobar[i].hight        = 0;
  pisobar[i].epaisseur    = 0;
  pisobar[i].isoblock     = 0;
  if (type == 1 || type == 3)
    pisobar[i].cj[0]       = '\0';
  else if (type == 2)
    strcpy(pisobar[i].cj,"PL");
  else
    pisobar[i].cj[0]       = '\0';
  pisobar[i].reference[0] = '\0';
  if (type == 1 || type == 3)
    pisobar[i].finition[0]  = '\0';
  else if (type == 2)
    strcpy(pisobar[i].finition,"***");
  else if (type == 3)
    pisobar[i].finition[0] = '\0';
  pisobar[i].haut = 0;
}
/*****************************************************************************/
int pop_up(it,x,y,isol_unit)
int it,x,y,isol_unit;

#define BXO bdata[it].bx + bdata[it].dx + x
#define BYO bdata[it].by + y

{
 int erreur,num_err,ok,op;
 float limite;
 char *choice;
 char string[50];
```

```
switch(it)
 {

   case 5:
    /* if strcmp(pbu[bld_unit].wexptyp,...) == 0 */
    if (type == 1)
       erreur = get_choice(BXO,BYO,epaisseur,&choice,0,1);
    else if (type == 2)
       erreur = get_choice(BXO,BYO,typebardint,&choice,0,1);
    break;

   case 6 :
    if (type == 1)
       erreur = get_choice(BXO,BYO,cjtab,&choice,0,1);
    else if (type == 2)
       erreur = get_choice(BXO,BYO,finbardint,&choice,0,1);
    break;

   case 7 :
    if (type == 1)
        erreur = get_choice(BXO,BYO,parevapeur,&choice,0,1);
    break;

   case 8 :
    if (type == 1)
        erreur = get_choice(BXO,BYO,isoblock,&choice,0,1);
    break;

   case 20 : /* seulement dans le cas de l'isolation */
    erreur = get_choice(0,60,select,&choice,0,1);
    if (strcmp(choice,"  TOITURE   ") == 0)
       return(1);
    if (strcmp(choice,"  MURS      ") == 0) return(0);
    if (strcmp(choice,"  TOITURE   ") != 0 &&
        strcmp(choice,"  MURS      ") != 0)
       return(-1);


   default :
    return(0);
 }
if (erreur == -1) return(-1);
decode_value(choice,it,isol_unit);
encode_value(string,it,isol_unit);
init_drawbox(x,y);
draw_box(it,bdata,cwbox,cwtx,string);
```

```
ok = validate_isol(it,isol_unit,&num_err,&limite,1);
if (ok != 0) message_err(num_err,7,0.);
return(1);
}
/*****************************************************************************/
/* Fonction de mise a jours des donnees.                                     */
/* Une augmentation du niveau implique une diminution de la hauteur, si
*/
/* celle ci n'est pas maximle (jusque toiture).                              */
/* une augmentation de la distance implique une diminution de la longueur
*/
/* si celle si n'est pas maximale.                                           */
/*****************************************************************************/
int maj_add(it,i,side)
int i,*it;
char side;
{
 char string[50];
 float H,s1,lenght,lim,A,B,x1,x2;

 float T1 = tan((double)(pbu[bld_unit].sll / 100));
 float T2 = tan((double)(pbu[bld_unit].slr / 100));
 float H1 = pbu[bld_unit].ehl;
 float H2 = pbu[bld_unit].ehr;
 int  dim,j;
 A = (float)min(H1,H2);
 B = (float)max(H1,H2);
 if (T1 > 0) x1 = (pisobar[i].level - H1) / T1;
 else  x1 = 0;
 if (T2 > 0) x2 = pbu[bld_unit].span - (pisobar[i].level - H2);
 else x2 = pbu[bld_unit].span;
 if (bld_unit > 0 )
  {
    if ((side == 'F' && pbu[bld_unit].sign == +1) ||
        (side == 'B' && pbu[bld_unit].sign == -1))
          T2 = 0;
    else if ((side == 'F' && pbu[bld_unit].sign == -1) ||
             (side == 'B' && pbu[bld_unit].sign == +1))
      {
       T2 = T1;
       T1 = 0;
      }
  }
 if (bld_unit == 0)
    s1 = (H2 - H1 + pbu[bld_unit].span * T2) / (T1+T2);
 else if (bld_unit > 0 && T1 > 0)
```

```
    s1 = (2 * pbu[bld_unit].span * T1) / (T1+T1);
else if (bld_unit > 0 && T2 > 0)
    s1 = (2 * pbu[bld_unit].span * T2) / (T2+T2);
if (bld_unit == 0)
 dim = pbu[bld_unit].nbrtrv;
else
 dim = pbu[bld_unit].tobay - pbu[bld_unit].frbay + 1;
  lenght = 0;
for (j = 0 ; j < dim ;j++)
  lenght += pbs[pbu[bld_unit].bssw].bs[j];
if (side == 'F' || side == 'B')
 {
  if ((bld_unit == 0) || (bld_unit > 0 && T1 > 0))
      H = T1 * s1 + H1;
  else if (bld_unit > 0 && T2 > 0)
      H = T2 * pbu[bld_unit].span + H1;
 }
else if (side == 'L' || side == 'R' || side == 'X')
 {
  if (side == 'L')   H = pbu[bld_unit].ehl;
  else   H = pbu[bld_unit].ehr;
 }
else  if (side == 'T')
  H = pbu[bld_unit].span;

if (pisobar[i].distance - s1 > 0)
lim = (pisobar[i].lenght - s1) * T2 ;
else lim = (s1 - pisobar[i].distance) * T1;

switch(*it)
 {
  case 1 : /* niveau modifie */
    if (type == 1 || type == 2)
     {
     /* maj de la hauteur*/
     float limite;
      int num_er,ok;
      if (side == 'L' || side == 'R' || side == 'X' || side == 'T')
       {
          if (pisobar[i].hight + pisobar[i].level > H)
           {
            pisobar[i].hight = H - pisobar[i].level;
            sound_bell();
            encode_value(string,4,i);
            draw_box(4,bdata,cwbox,cwtx,string);
           }
```

```
        }
      else if ((side == 'F' || side == 'B') && pisobar[i].haut == 0)
        {
        if ((side == 'F' || side == 'B') && action == 3)
            {
              if (T2 > 0 && pisobar[i].level + pisobar[i].hight  > lim  &&
pisobar[i].haut == 0)
                {
                   sound_bell();
                   pisobar[i].hight = H - pisobar[i].level - lim ;
                   encode_value(string,*it,i);
                   draw_box(*it,bdata,cwbox,cwtx,string);
                   message_err(8,RED,0.);
                   j = 2;
                   maj_add(&j,i,side);
                }
            }
        }
      else if ((side == 'F' || side == 'B') && pisobar[i].haut == 1 && action
== 3)
        {
           strcpy(string,"0");
           decode_value(string,4,i);
           encode_value(string,4,i);
           init_drawbox(x,y);
           draw_box(4,bdata,cwbox,cwtx,string);
        }
      if (side == 'F' || side == 'B')
        {
           if (H - pisobar[i].level < A)
            {
              if (pisobar[i].distance < x1) pisobar[i].distance = x1;
              else if (pisobar[i].distance > x2) pisobar[i].distance = x2;
              encode_value(string,2,i);
              decode_value(string,2,i);
              encode_value(string,2,i);
              init_drawbox(x,y);
              draw_box(2,bdata,cwbox,cwtx,string);
              ok = validate_isol(it,isol_unit,&num_err,&limite,3);
            }
           else if (H - pisobar[i].level > A && H - pisobar[i].level < B)
            {
              if (H1 < H2)
                {
                   if (pisobar[i].distance < x1)
                          pisobar[i].distance = x1;
```

```
                    else if (pisobar[i].distance > pbu[bld_unit].span)
                      pisobar[i].distance = pbu[bld_unit].span;
                  }
                 else if (H1 > H2)
                  {
                    if (pisobar[i].distance > x2)
                          pisobar[i].distance = x2;
                    else if (pisobar[i].distance < 0)
                       pisobar[i].distance = 0;
                  }
               encode_value(string,2,i);
                 decode_value(string,2,i);
                 encode_value(string,2,i);
                 init_drawbox(x,y);
                 draw_box(2,bdata,cwbox,cwtx,string);
               }
              ok = validate_isol(3,i,&num_err,&limite,3);

              if (ok == 0)
               {
                 pisobar[i].lenght = limite;
                 encode_value(string,3,i);
                 init_drawbox(x,y);
                 draw_box(3,bdata,cwbox,cwtx,string);
               }
            }
         }
        else if (type == 3)
         {
           j = 2;
           maj_add(&j,i,side);
         }
        break;

case 2 : /* longueur */
  if ((*it == 1 && type == 3) ||
      ((*it == 2 || *it == 1) && (type == 1 || type == 2)))
     {
       if ((side == 'F' || side == 'B') && T2 > 0 && pisobar[i].haut == 0)
         {
           float k;
           k = (H - pisobar[i].hight - pisobar[i].level) / T2  + s1;
           if (pisobar[i].distance + pisobar[i].lenght > k )
              {
                pisobar[i].lenght = k - pisobar[i].distance;
                init_drawbox(x,y);
```

```
                if (type == 1 || type == 2)
                 {
                  encode_value(string,3,isol_unit);
                  draw_box(3,bdata,cwbox,cwtx,string);
                 }
                else
                 {
                  encode_value(string,2,isol_unit);
                  draw_box(2,bdata,cwbox,cwtx,string);
                 }
              }
            }
          }
     break;

  case 3 :
    if (type == 1 || type == 2) break;
    else if (type == 3 && pisobar[i].haut == 0)
     {
       j = 2;
       maj_add(&j,i,side);
     }
    break;
  case 4 :
    if (type == 3) break;
    else if ((type == 1 || type == 2 ) && pisobar[i].haut == 0)
     {
       j = 2;
       maj_add(&j,i,side);
     }
   break;
  }
}

/*****************************************************************************/
message_err(num_err,color,limite)
int color;
int num_err;
float limite;

{
 int bkcol,frcol;
 char buffer[60];
 bkcol = getpixel(1,430);
 frcol = getcolor();
 setfillstyle(1,bkcol);
```

```
bar(1,430,638,450);
if (!color) return;
setcolor(color);
switch(num_err)
{
case 1 :
    outtextxy(110,435,"Cette valeur est obligatoire");
    break;

case 2 :
  sprintf(buffer,"Le niveau doit etre inferieur a %7.3f",limite);
  outtextxy(110,435,buffer);
  break;

case 3 :
    sprintf(buffer,"La hauteur ne doit pas depasser %7.3f",limite);
    outtextxy(110,435,buffer);
    break;

case 4 :
    sprintf(buffer,"La distance ne doit pas depasser : %7.3f",limite);
    outtextxy(110,435,buffer);
    break;

case 5 :
    sprintf(buffer,"La longueur ne doit pas depasser %7.3f",limite);
    outtextxy(110,435,buffer);
    break;

case 6 :
    sprintf(buffer,"La hauteur ne doit pas depasser %6.2f",limite);
    outtextxy(110,435,buffer);
    break;

case 7 :
    outtextxy(110,435,"        choisir un cote svp  ");
    break;

case 8 :
    outtextxy(110,435," La hauteur est mise a jour ");
    break;

case 9 :
    outtextxy(110,435," choisir d'abord un cote svp  ");
    break;
```

```
 case 10 :
     outtextxy(80,435," Choisir un cote du batiment principal ou d'un
appentis");
     break;
 case 11 :
     if (type == 1) outtextxy(110,435,"Selectionner une unite d'isolation svp
");
     else if (type == 2) outtextxy(110,435,"Selectionner une unite de
bardage svp ");
     else if (type == 3) outtextxy(110,435,"selectionner une unite de
sous_basement svp ");
     break;
 case 12 :
    if (type == 1) outtextxy(110,435,"Aucune partie de ce cote n'est isolee
");
    else if (type == 2) outtextxy(110,435,"Pas de bardage sur ce cote ");
    else if (type == 3) outtextxy(110,435,"Pas de sous-basement sur ce cote
");
    break;
 case 13 :
    outtextxy(110,435,"Selectionner un batiment ");
    break;
 case 14 :
    outtextxy(110,435," La longueur est mise a jour");
    break;
 case 15 :
    sprintf(buffer," la distance doit etre superieur a %7.3f",limite);
    outtextxy(110,435,buffer);
  case 16 :
    outtextxy(110,435," La distance est mise a jour");
    break;

    break;
  }
 }
```